NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

FEASIBILITY STUDY OF A VISION-BASED LANDING SYSTEM FOR UNMANNED FIXED-WING AIRCRAFT

by

Tyler B. McCarthy

June 2017

Thesis Advisor: Oleg A. Yakimenko
Second Reader: Fotis A. Papoulias

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2017	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
FEASIBILITY STUDY OF A VISION-BASED LANDING SYSTEM FOR UNMANNED FIXED-WING AIRCRAFT

5. FUNDING NUMBERS

6. AUTHOR(S) Tyler B. McCarthy

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (maximum 200 words)

Successful landing of an autonomous unmanned aerial vehicle requires a high degree of accuracy and efficient, real-time processing. This research applies systems engineering concepts to investigate the feasibility of applying computer vision techniques and visual feedback in the control loop for an autonomous system. This thesis examines the framework and performance of an algorithm designed to detect and track a runway in images captured from a camera onboard an aircraft during the final approach and landing stages of flight. Using a series of image processing techniques to localize the runway and the Hough transformation for line detection, the algorithm is capable of detecting the edges of a runway with over 96 percent accuracy through 3000 test images. The operating conditions for this algorithm include any scenario in which visual flight rules apply. Additionally, the system will perform with runways that comply with Federal Aviation Administration regulations. Future applications of this algorithm should include aircraft attitude and pose estimation as well as full integration into an autonomous aircraft control system.

14. SUBJECT TERMS autonomous systems, auto-land, computer vision, image processing, unmanned aerial vehicles	**15. NUMBER OF PAGES** 93
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

FEASIBILITY STUDY OF A VISION-BASED LANDING SYSTEM FOR UNMANNED FIXED-WING AIRCRAFT

Tyler B. McCarthy
Ensign, United States Navy
B.S., United States Naval Academy, 2016

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 2017

Approved by: Oleg A. Yakimenko
Thesis Advisor

Fotis A. Papoulias
Second Reader

Ronald E. Giachetti
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Successful landing of an autonomous unmanned aerial vehicle requires a high degree of accuracy and efficient, real-time processing. This research applies systems engineering concepts to investigate the feasibility of applying computer vision techniques and visual feedback in the control loop for an autonomous system. This thesis examines the framework and performance of an algorithm designed to detect and track a runway in images captured from a camera onboard an aircraft during the final approach and landing stages of flight. Using a series of image processing techniques to localize the runway and the Hough transformation for line detection, the algorithm is capable of detecting the edges of a runway with over 96 percent accuracy through 3000 test images. The operating conditions for this algorithm include any scenario in which visual flight rules apply. Additionally, the system will perform with runways that comply with Federal Aviation Administration regulations. Future applications of this algorithm should include aircraft attitude and pose estimation as well as full integration into an autonomous aircraft control system.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

DGPS	differential global positioning system
DME	distance measuring equipment
DOD	Department of Defense
FAA	Federal Aviation Administration
FLOLS	Fresnel lens optical landing system
HSV	hue, saturation, value
ILS	instrument landing system
INS	inertial navigation system
JPALS	joint precision approach and landing system
RGB	red, green, blue
UAS	unmanned aerial system
UAV	unmanned aerial vehicle
VFR	visual flight rules

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Research concerning autonomous vehicles and computer vision has expanded rapidly, and the two fields are closely related. The use of autonomous unmanned aerial vehicles (UAV) by the Department of Defense (DOD) has become a vital component of national security with the most recent successes demonstrated by the RQ-4 Global Hawk and the X-47B. Both of these aircraft are capable of autonomous takeoff, mission execution, and landing. These systems use an intricate differential global positioning system (DGPS) to collect accurate position estimates that allow for successful landings. This research follows a systems engineering approach to evaluating the feasibility of using computer vision and visual feedback techniques to control the aircraft in the final approach and landing stages of flight. Specifically, this evaluates framework for an algorithm to detect and track a runway from a camera onboard an aircraft. This thesis offers an improvement to current vision-based runway detection frameworks such as those made relevant by Shang and Zhongke (2011) in their paper titled, "Vision-Based Runway Recognition for Autonomous Landing of a UAV."

The first stages of image processing and detection use common pre-processing techniques, the first being thresholding. While thresholding is usually applied to the intensity of grayscale images, this research favors thresholding in the hue, saturation, and value (HSV) color space. This allows the algorithm to localize the position of the runway using chromaticity information as well as intensity information. The additional information improves the performance of the algorithm and results in improved isolation of the runway area and elimination of noise and clutter. The algorithm uses a hue threshold between 0 and 0.440 and between 0.700 and 1.0 to eliminate color ranges in the blue spectrum that easily coincide with the color of the sky. Additionally, the saturation threshold is set between 0 and 0.2 and the value threshold is set between 0.637 and 1.0 to capture the lighter and brighter runway markings.

Following HSV thresholding, a series of morphological operations are applied to the image to eliminate clutter and noise while ensuring the entirety of the runway area is captured. The first of the basic operations used is dilation, which increases the perimeter

of a binary region in order to capture lost portions of the area within the region of interest (MathWorks 2017). The second basic operation is dilation, which decreases the perimeter of a binary region in order to eliminate clutter or unwanted areas (MathWorks 2017). Together, these operations form the basis of all morphological tasks. The algorithm first applies a filling operation, which dilates then erodes the image to fill holes in a region and recapture lost information. The algorithm then applies an opening operation to eliminate the noise and clutter in the image, specifically any grouping of pixels with a perimeter smaller than 30 pixels. Finally, the image is dilated with a vertical, rectangular structuring element to increase any portions of the runway region that may have been lost in the previous operations. Upon completion, the algorithm applies a second filling operation to ensure the remaining regions contain no unwanted gaps or holes. The result of these morphological operations is a masked image that localizes the runway area and eliminates as much noise and irrelevant area as possible.

To reduce noise in the image resulting from internal camera operations or the environment, the algorithm filters the masked image to reduce the effects of Gaussian, salt and pepper, and quantization noise. The median filter was chosen instead of the mean filter or other techniques due to its robustness and edge preservation characteristics. As Dangeti (2003, 18) describes in her research, median filtering samples a predetermined window of pixels from the image and replaces the center pixel of interest with the median intensity value.

In an additional attempt to preserve edges within each frame, the algorithm sharpens the image using MATLAB's "imsharpen" function with an edge radius of one pixel. The algorithm uses a particular image sharpening technique, called unsharp masking, which creates a Gaussian-blurred version of the original image and subtracts the blurred copy from the original, resulting in sharper edges in areas that meet the conditions of the system (Cambridge in Colour 2017). This process advantageously produces edges that are sharply delineated and easier to detect in later stages of the algorithm.

With a sharpened image, the pre-processing steps are complete and the algorithm can successfully apply edge detection techniques. This research only compared

traditional edge detection methods because they are more computationally efficient while maintaining a sufficient level of accuracy. In a comparison of the Sobel method and the Canny method, the Sobel operator was the edge detection method of choice due to its lesser processing time. The Sobel method uses a 3x3 operator that sums the gradient values of orthogonal vectors resulting in a magnitude and direction value for a given neighborhood of pixels (Sobel 2014). Within the algorithm, the Sobel operator captures all edges within the image that meet a minimum threshold of 0.015, which was determined through trial and error. The result is a binary image that only displays the lines meeting the threshold set by the Sobel operator.

The application of the Hough transformation is the most important component of the runway detection algorithm. Paul Hough first patented the Hough transform in 1962 as a means to detect patterns created by subatomic particles in a bubble chamber. The Hough transform, as used today, transforms all points in the image frame in a new polar parameter space. Within the parameter space, all points in an image frame correspond to a sinusoid, points in the parameter space are straight lines in the image, and points on the same line in the image frame will share a common point in the parameter space (Hart and Duda 1972, 4). The algorithm applies the Hough transform to the edge-detected image and searches for lines within the range of -35 and 35 degrees to match the expected runway angle from the final approach perspective. The top 5–7 peaks in the Hough parameter space are chosen and analyzed in order to determine the best match for the runway edges. Typically, the top two peaks are the most likely matches; however, the most prominent lines in the image will not necessarily coincide with the edges of the runway. To avoid incorrect identification, the algorithm examines the resulting lines for a number of common error scenarios. The first error check is the distance between points on the runway edges. Any separation of less than 20 pixels indicates that the detected runway edges are too close and that incorrect lines have been selected. Therefore, the algorithm analyzes the next grouping of peaks in the Hough space. Conversely, if the distance between runway endpoints is greater than 150 pixels in the x direction or 40 pixels in the y direction, the detected edges are too far apart and the algorithm signals a line mismatch. Together, these test scenarios ensure that the detected edges fall within the

expected range for the runway geometry. Additionally, in real-time application, the detected runway edges are averaged between image frames, achieving the effect of a low pass filter. This limits the effect of mechanical vibrations on the camera and it reduces the impact of slight errors within the algorithm.

The algorithm was tested on 3000 approach images to Monterey Regional Airport taken from a TASE200 camera attached to the airframe of a Cessna 206 below the left wing. Overall, the system achieved an accuracy of 96.2 percent across all images, incorrectly identifying only 114 of the 3000 frames. The primary sources of inaccuracies were distance extremes and misidentified clutter within the image. The algorithm performs comparatively poorly at extremely long distances and extremely short distances. At long distance, noise and blurring in the image make accurate detection of runway markings difficult. At short distances, the aircraft is at very low altitudes, which distorts the perspective of the runway and makes accurate detection more difficult. Noise and clutter only become a problem in images when they align with one of the runway edges. If clutter is close enough to the runway edge and large enough to pass through noise and clutter elimination methods, it can be misidentified as a part of the runway edge. Outside of these specific scenarios, the algorithm performs well. An original runway approach image and its corresponding output from the algorithm are shown in Figure 1.

Figure 1. Original Approach Image (left) and Algorithm Output (right)

Future applications of this algorithm could include attitude and pose estimation as well as complete integration with an aircraft control system. Based on the performance of the runway detection and tracking algorithm, it is evident that computer vision could play a significant role in the navigation and control of autonomous UAVs. However, much

work remains to integrate the algorithm with an aircraft's control system. For one, attitude and pose estimation must be applied to accurately estimate the position of the aircraft. This information will play a crucial role in fulfilling the feedback control loop within the control system of the aircraft. Overall, it is reasonable to expect that visual feedback can play an important role in the future of autonomous aircraft.

References

Cambridge in Colour. 2017. "Sharpening using an Unsharp Mask." Accessed April 19, 2017. http://www.cambridgeincolour.com/tutorials/unsharp-mask.htm.

Dangeti, Sarita Veera. 2003. "Denoising Techniques—A Comparison." Master's thesis, Louisiana State University. Accessed April 12, 2017. http://etd.lsu.edu/docs/available/etd-1219102-152426/unrestricted/Dangeti_thesis.pdf.

Duda, Richard O., and Peter E. Hart. 1971. "Use of the Hough Transformation to Detect Lines and Curves in Pictures." *Communications of the ACM* 15, no. 1 (January): 11–15. Accessed 17 April, 2017. http://www.dtic.mil/docs/citations/ADA457992.

Hough, P. V. C. 1962. Method and Means for Recognizing Complex Patterns. U.S. Patent 3,069,654, filed March 25, 1960, and issued December 18, 1962.

MathWorks. 2017. "Morphological Dilation and Erosion." Accessed May 15, 2017. The MathWorks Inc. https://www.mathworks.com/help/images/ref/imfill.html.

Shang, Jiajia, and Zhongke Shi. 2007. "Vision-Based Runway Recognition for Autonomous UAV Landing." *IJCSNS International Journal of Computer Science and Network Security* 7 no. 3: 112–117. Accessed April 7, 2017. http://www.sciencedirect.com/science/article/pii/S1877705812021844.

Sobel, I., and G. Feldman. 1968. "An Isotropic 3x3 Image Gradient Operator." Lecture, Stanford Artificial Intelligence Project. Accessed April 19, 2017. https://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would first like to thank my mom, dad, brother, and sister for their unwavering support, no matter the time or place. I would also like to thank all of my family and friends for making the journey exciting and worthwhile along the way.

Special recognition goes to my advisor, Professor Oleg Yakimenko, for his steady guidance and unending stream of knowledge to help me complete this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The introduction presents the current state of unmanned aerial vehicles (UAV) and auto-land procedures. This chapter will describe UAVs in a Department of Defense (DOD) context while linking the goals of this research to DOD requirements for autonomous systems. Additionally, the overall concept and objectives for a vision-based landing system and the scope and limitations involved in the research are outlined.

A. VISION FOR UNMANNED AERIAL VEHICLES

Autonomous systems are fulfilling increasingly important and complex roles within the DOD and foreign militaries across the globe. Not only are autonomous systems capable of taking the warfighter out of dangerous situations, but they can operate in diverse environments while accomplishing a versatile collection of missions. The DOD *Unmanned Systems Integrated Roadmap for FY2011–2036* (2011, 3) states, "DOD envisions unmanned systems seamlessly operating with manned systems while gradually reducing the degree of human control and decision making required for the unmanned portion of the force structure." Thus, the ideal autonomous system is one that can accomplish its mission without any human intervention by making tactical or strategic decisions without a human in the control loop. While the capabilities of unmanned aerial systems (UAS) are increasing, they still require significant levels of human oversight and external control inputs, usually via remote piloting at ground control stations. In order to achieve increased autonomy in DOD systems, decision-making and control processes must be integrated into the system itself. The advanced autonomous vehicle should be a self-contained system capable of gathering information from its surroundings, processing the information relative to its tactical scenario and mission, determining a course of action, and acting on its decision without external human inputs.

The *DOD Research and Engineering Technical Assessment on Autonomy* (Office of Technical Intelligence 2015, 4) encourages additional research into low-cost systems capable of achieving autonomous "perception, cognition, and action." The statement implies an underlying intent to develop unmanned systems that fulfill a variety of

dangerous and complex roles while doing so at a lower cost than current mission execution levels. Achieving full autonomy in aerial vehicles is technologically challenging and the solutions tend to be expensive. This research offers a method to increase autonomy in UAVs via low-cost vision systems. While this research will focus on aerial systems, the basic principles of vision-based control are applicable to all autonomous systems in the DOD.

B. CURRENT STATE OF MANNED AND UNMANNED AIRCRAFT LANDING PROCEDURES

This section provides a brief overview of both manned and unmanned aircraft landing procedures. The first subsection focuses on general landing procedures while the second and third subsections focus on the technical aspects of assisted and autonomous landings.

1. General Landing Procedures

This research focuses on the stages of landing from final approach to touchdown of the aircraft. Specific landing procedures will vary based on the size and aerodynamic characteristics of the aircraft and environmental conditions; however, the Federal Aviation Administration outlines a general approach to landing in its *Airplane Flying Handbook*. The pilot reaches the final approach when the heading of the aircraft, given no crosswind, is aligned with the center of the runway and the aircraft is following the desired glide slope (Federal Aviation Administration [FAA] 2016a, 8.3). Throughout the final approach, the pilot should make constant adjustments to align the aircraft with the runway centerline, maintain the appropriate glide slope, and maintain the appropriate airspeed via control inputs to the rudder, ailerons, elevators, flaps, and engine power (FAA 2016a, 8.3-8.6). The pilot should aim for a predetermined touchdown point throughout the final approach, typically delineated by two specific markings on either side of the runway beyond the threshold (FAA 2016a, 8.10). A useful tool for the pilot in achieving a consistent approach angle is to ensure that runway perspective remains the same (FAA 2016a, 8.10). An elongated, narrow perspective indicates a steeper glide slope while a flatter, shorter perspective indicates a shallow glide slope (FAA 2016a,

8.10). This technique should be continuously applied until the pilot must flare the nose of the aircraft and touchdown. The common theme throughout the landing process is that visual cues are essential to maintaining an accurate final approach and completing a successful landing. By using a vision-based landing procedure for autonomous systems, the same general procedure applies.

2. Assisted Landings

A common and well-accepted landing navigation aid is the Instrument Landing System (ILS). The system uses components called a localizer and a glide slope to provide incoming aircraft with azimuth and elevation information for landing (FAA 2016b). The localizer generates VHF signals that the incoming aircraft can translate into relative positions left or right of the runway centerline (FAA 2016b). The glide slope station also generates VHF signals that the incoming aircraft translates into elevation positions either above or below a three-degree angle of descent (FAA 2016b). When carefully integrated with runway markings and lighting, ILS becomes an integral tool for pilots.

Distance Measuring Equipment (DME) is another navigational aid that is often integrated with ILS. DME stations at airports receive and relay radio signals from incoming aircraft, which the aircraft's DME equipment can then use to calculate distance from the runway (FAA 2014). DME differs from localizer and glide slope stations because it provides direct slant range between the runway and the aircraft rather than azimuth and elevation information (FAA 2014).

Naval aviators use an improved version of the Fresnel Lens Optical Landing System (FLOLS) as a navigation aid for landing onboard aircraft carriers (Golovcsenko 1976, 9–11). As with ILS, FLOLS provides the incoming aircraft with its position relative to the center of the runway through a lighted reference station located at the edge of the flight deck (Golovcsenko 1976, 9). The pilot sees a bar of light at the center of the FLOLS station, when the bar is stationed above the green reference lights on either side, the glide slope is too steep and when the bar is stationed below, the glideslope is too shallow (Golovcsenko 1976, 9). The FLOLS is depicted in Figure 1.

Figure 1. FLOLS System Onboard an Aircraft Carrier. Source: NAVAIR Fleet Readiness Center Southwest (2003).

3. Current Autonomous Landing Capability

Few unmanned aerial vehicles can successfully takeoff and land autonomously. Most notable are the RQ-4 Global Hawk and the X-47B, both produced by Northrop Grumman. The RQ-4 Global Hawk is an operational unmanned aerial system designed for intelligence, surveillance, and reconnaissance with over 200,000 operational flight hours (Northrop Grumman 2017). The X-47B is an unmanned combat air system (UCAS) created to test the feasibility of an autonomous carrier-based air platform (Northrop Grumman 2015). Together, these aircraft represent the future of UAVs as used by the U.S. military while leading the field in auto-land technology.

The autonomous landing process for both aircraft is similar. Both use differential global positioning systems (DGPS) and inertial navigation systems (INS) to make precision adjustments in the landing phase. DGPS uses a stationary radio transmitter in addition to the four satellites necessary for a GPS fix, significantly increasing the accuracy of the system (Defense Standardization Program 2010, 5). The Global Hawk specifically uses dual KN-4072 INS/GPS onboard systems for guidance in the landing stage (Loegering and Harris 2002, 2). Prior to takeoff and mission execution, the Global

Hawk is loaded with a series of GPS waypoints for the landing sequence that guide the aircraft to the runway and indicate when the aircraft should begin executing specific landing procedures (Loegering and Harris 2002, 3). For example, a waypoint positioned approximately 150 seconds from the point of touchdown indicates that the Global Hawk should change speed to meet final approach requirements (Loegering and Harris 2002, 3). Though not an exact replication, the X-47B utilizes a similar procedure that closely mirrors the Joint Precision Approach and Landing System (JPALS). JPALS sea-based systems, such as those used on aircraft carriers, implement a landing process similar to that of the Global Hawk, but they also integrate data from a shipboard INS to account for the continuous motion of the ship (Defense Standardization Program 2010, 6). This additional layer of INS data allows incoming aircraft such as the X-47B to land with the accuracy and precision required for carrier-based operations. DGPS-based systems, such as those found on the RQ-4 and the X-47B, offer superior performance for autonomous landings, but they add a significant level of complexity to the system.

C. VISION-BASED LANDING SYSTEM CONCEPT

Developing a vision-based landing system requires three central components: an image processing algorithm, a pose estimation algorithm, and a control system integrated with the unmanned aerial vehicle.

1. Control

The execution of the vision-based landing system will follow the framework for a standard control loop, as shown in Figure 2. Error signals related to heading, altitude, and speed will serve as inputs to the controller, which calculates corresponding outputs for the aircraft control surfaces to achieve the desired state. In this case, the desired state is a heading aligned with the center of the runway, a three-degree angle of descent, and consistent approach speed. The resulting state of the aircraft following these control actions will be measured and fed back into the controller via the vision-based system established in the previous steps. The camera will capture an image of the runway, which the pose estimation algorithm will then use to estimate the attitude and position of the aircraft and consequently the error signal for the controller. The unmanned system will

continuously execute the control loop until the aircraft safely lands. In Figure 2, e_a, e_h, and e_s represent the altitude, heading, and speed error, respectively. Likewise, δ_e, δ_r, and δ_t represent the required changes in elevator angle, rudder angle, and thrust, respectively.

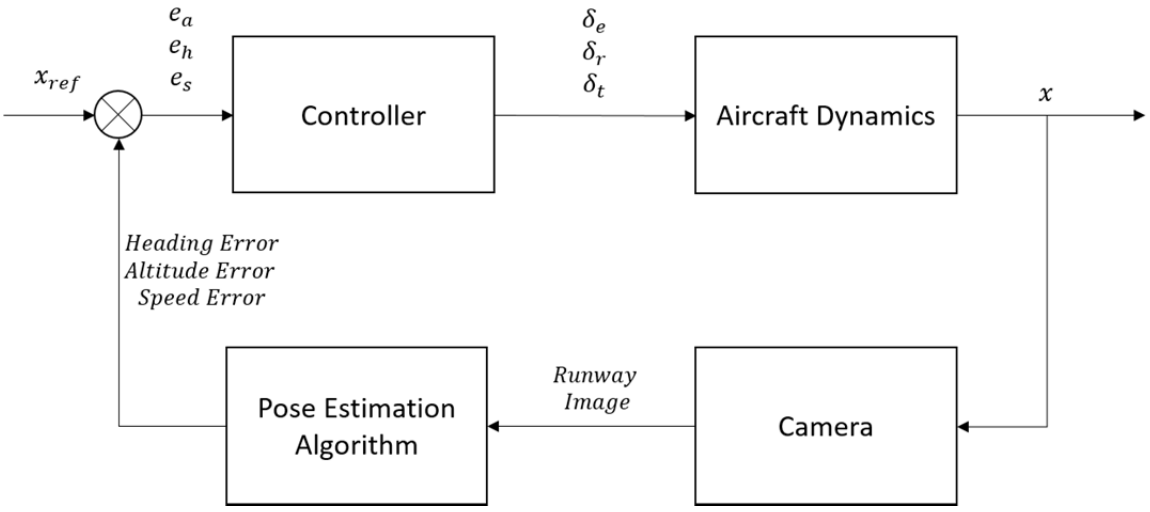


Figure 2. Vision-Based Landing Control System

2. Image Processing

The image processing component of the vision-based landing system involves detection and tracking of the runway in images taken from a camera onboard the aircraft. The objective of the image processing component is to accurately identify and localize the runway while distinguishing relevant markings and characteristics of the runway. The various computer vision techniques that allow for the successful completion of this step will be outlined in later chapters, as the image processing component is the focus of this research.

3. Pose and Attitude Estimation

Pose and attitude estimation is the process of calculating the roll, pitch, yaw, glide slope, and speed values for the aircraft. This is the stage in the vision-based landing system in which the system gains situational awareness by analyzing runway information provided by the previous image processing algorithm. Thus, the same visual cues used by human pilots can be used by unmanned autonomous systems.

a. Heading Error

Heading error is crucial to maintaining alignment with the centerline of the runway in the final approach stage. The system can maintain proper heading by ensuring that the internal angles of each edge of the runway are equal, assuming that the position of the camera in relation to the center of the aircraft is known and that the aircraft is not experiencing any sideslip. Equation 1.1 describes the relationship between internal runway edge angles and the heading error.

$$e_h = \theta_l - \theta_r + f(\beta) \tag{1.1}$$

In Equation 1.1, the theta values represent the left and right internal angles, respectively, while the final value represents a function that adjusts for the sideslip angle of the aircraft depending on the aircraft dynamics and external conditions.

b. Altitude Error

The algorithm should calculate altitude error as a ratio of the distant runway edge to the leading runway edge, assuming that the length and width of the runway is known. If the aircraft is maintaining a consistent glideslope through descent, the ratio will remain the same. Larger or smaller ratios between the runway ends should trigger an error signal in the system, as these conditions indicate the glide slope is either too shallow or too steep.

c. Speed Error

Estimating speed from vision alone is a difficult task and it is not one that human pilots can accurately complete. Therefore, it is reasonable to assume that some assistance may be required from additional sensors, such as pitot tubes, in order to calculate airspeed. Visual cues that may lend to the calculation of airspeed could include the calculation of the rate of change of a known distance in the image frame, such as the width of the runway.

D. RESEARCH QUESTION AND OBJECTIVES

A capability need statement for future UAVs undoubtedly evokes the need to create an accurate, robust, and cost efficient system that can successfully achieve a given mission profile without human intervention. While achieving autonomy in aerial systems is a multi-faceted issue, developing an improved autonomous landing capability remains a critical issue for the future of unmanned systems. What is required is a systems engineering approach to the development and assessment of autonomous landing processes. While the RQ-4 and X-47B have demonstrated autonomous landing capabilities, they are complex, expensive, and still require some degree of human intervention. A critical component of systems engineering is the process of analyzing the solution space to find the best approach to overcome a given problem. The process must include trade-off and feasibility analysis to determine the best solution for all stakeholders. This thesis uses a systems engineering approach to evaluate the feasibility of a real-time, vision-based runway detection and tracking algorithm that will aid in the landing of a fixed-wing UAV. As this is a complex problem, it requires in-depth analysis and a fundamental understanding of the technical issues. This research must first explain the framework and methodology for the vision-based landing system, then conduct objective analysis of its performance and feasibility in future systems. Thus, this thesis aims to provide a quantitative and qualitative analysis of a computer vision approach intended to achieve an autonomous landing capability.

E. SCOPE, LIMITATIONS, AND ASSUMPTIONS

A satisfactory runway detection and tracking algorithm should be extremely accurate so as to prevent false detection and, therefore, the passing of erroneous signals to the aircraft control system. The algorithm should also be robust and reliable with the ability to detect the location of the runway at long and short ranges, in all conditions meeting visual flight rule (VFR) weather minimums as defined by the Federal Aviation Administration (FAA), and through noisy image frames taken from live video onboard the aircraft. Processing time must also be a consideration for eventual application in a real-time environment. Fast processing times are often at odds with increased accuracy

and robustness. Therefore, one must evaluate every aspect of the detection and tracking process based on necessity and optimization.

The fields of unmanned autonomous vehicle research and computer vision are vast, highlighting the importance of scope limitation for individual research projects. This thesis focuses on autonomous fixed-wing aircraft. Size of the aircraft is not necessarily a relevant factor so long as the characteristics of the aircraft do not call for special considerations for control during the final approach and landing stages of flight. Therefore, the results of the detection and tracking process for fixed-wing aircraft may still be relevant to all unmanned aerial vehicles including rotorcraft.

While the computer vision algorithm should be robust for use on a wide variety of runways, a few general assumptions are required. First, the runway markings used as visual cues should abide by Federal Aviation Administration standards as outlined in AC 150/5340-1L also titled "Standards for Airport Markings." Secondly, all runways should have a centerline and side stripes for accurate detection. These are standard markings for precision instrument runways. Lastly, there should be no obstacles or obstructions in line of sight to the runway in the final approach stage of flight. A runway meeting all of these assumptions is depicted in Figure 3.

Figure 3. Precision Instrument Runway and Markings in Accordance with Federal Aviation Administration Standards. Source: FAA (2015).

THIS PAGE INTENTIONALLY LEFT BLANK

II. DATA SAMPLES AND LITERATURE REVIEW

This chapter describes the data used for evaluation of the vision-based runway detection and tracking algorithm. A complete literature review and thesis overview is also included to provide reference frame for the reader.

A. DESCRIPTION OF AVAILABLE DATA

The images used for this research feature a final approach at Monterey Regional Airport (MRY). The data was collected from a Cessna 206 airplane with a TASE200 camera attached to the airframe below the left wing. The setup is shown in Figure 4.

Figure 4. Aircraft and Camera Setup for Collection of Data.
Source: Gloria (2016).

The camera has a frame rate of 30 frames per second (fps) and this research used 3000 frames of the final approach video. The remaining characteristics of the TASE200 camera are shown in Table 1.

Table 1. Specifications for TASE200 Camera. Adapted from Gloria (2016).

MECHANICAL SPECIFICATIONS	
Diameter	4.4 inches
Height	7.5 inches
Weight	2.34 pounds
PERFORMANCE	
Use	Daylight and infrared imaging
Pan limits	360˚ continuous
Tilt limits	+23˚/-203˚
IR camera	Resolution:640x480; HFOV: 10.5˚
Daylight camera	Optical zoom: 31x; HFOV:55.7˚-1.94˚

The images cover approximately one minute and 40 seconds of flight time in the final approach to the runway at Monterey Regional Airport. The starting altitude is approximately 1461.0 feet and the final altitude matches the elevation of Monterey Regional Airport's runway. The average airspeed throughout the descent was approximately 94.5 knots. Samples of the runway images from long, medium, and short range are shown in Figure 5.

Figure 5. Long (left), Medium (middle), and Short (right) Range Approach Images to Monterey Regional Airport

The TASE200 system also collected INS and GPS metadata from the flight that corresponds to each individual image frame. The GPS data from the final approach to Monterey Regional suggests a relatively straight flight path, as shown in Figure 6. The blue line represents the path of the aircraft while the red points represent the estimated target area of the camera throughout the final approach.

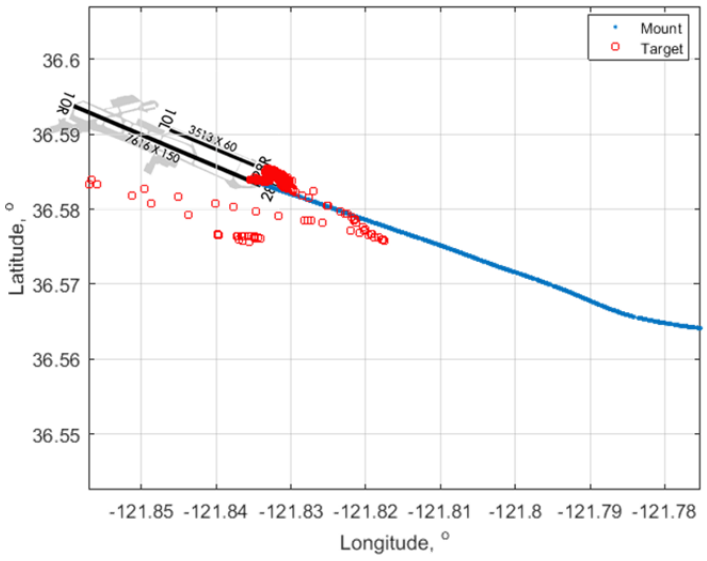


Figure 6. Flight Path Represented in Latitude and Longitude Coordinate Frame. Adapted from AirNav (2017).

The system also captured altitude, speed, and attitude information from the aircraft for each individual image frame. The altitude and heading data is shown in Figure 7. It is apparent that the aircraft follows a relatively stable rate of descent and maintains a consistent heading. In Figure 7, the blue line represents the aircraft altitude and the red line represents the altitude of the camera's target.

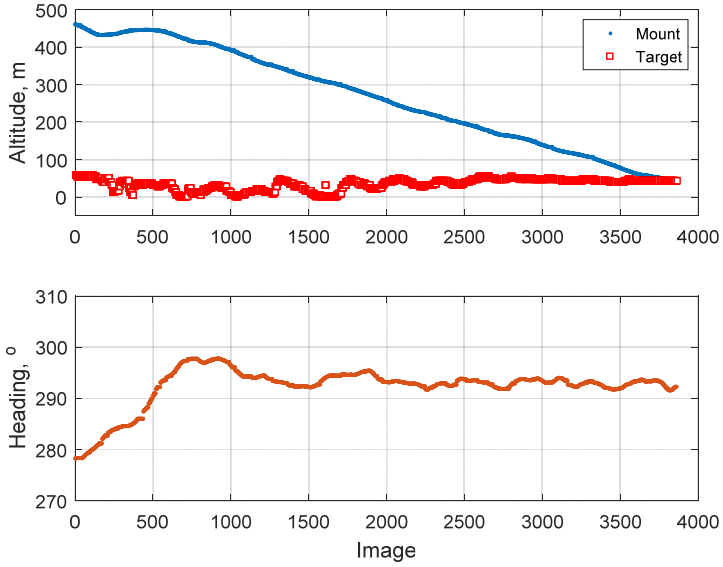


Figure 7. Altitude and Heading Flight Data

For future pose estimation evaluation, it will be important to consult the true roll, pitch, and yaw data from the flight. This data is displayed in Figure 8, where the blue and red lines represent the position of the TASE200 camera gimbal and mount, respectively.

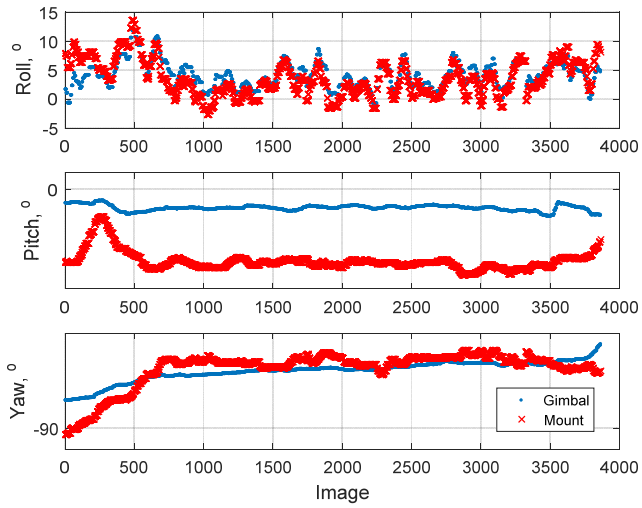


Figure 8. Roll, Pitch, and Yaw Flight Data

Additionally, the TASE200 system collected speed data from the flight. The corresponding information is shown in Figure 9. The top graph displays the aircraft speed calculated using GPS data while the bottom graph shows speed relative to the vertical axis of the body of the aircraft.

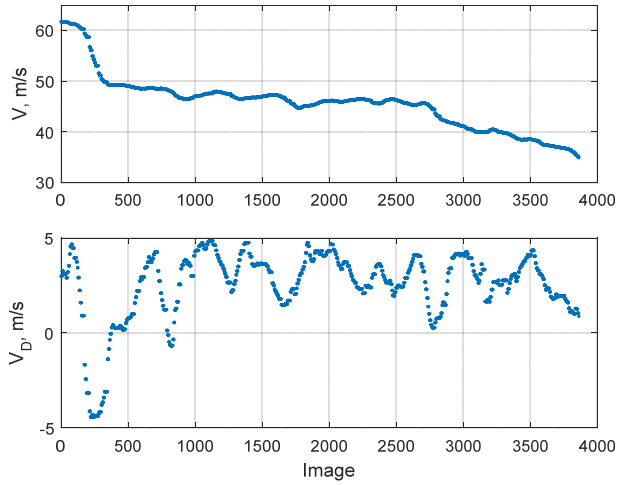


Figure 9. Speed Flight Data for GPS Speed (top) and Vertical Axis Speed (bottom)

Finally, it may be necessary to calculate transformations from the camera frame to the aircraft body frame or the world frame. These calculations would require information concerning the pose of the camera in each frame. This information is displayed in Figure 10.

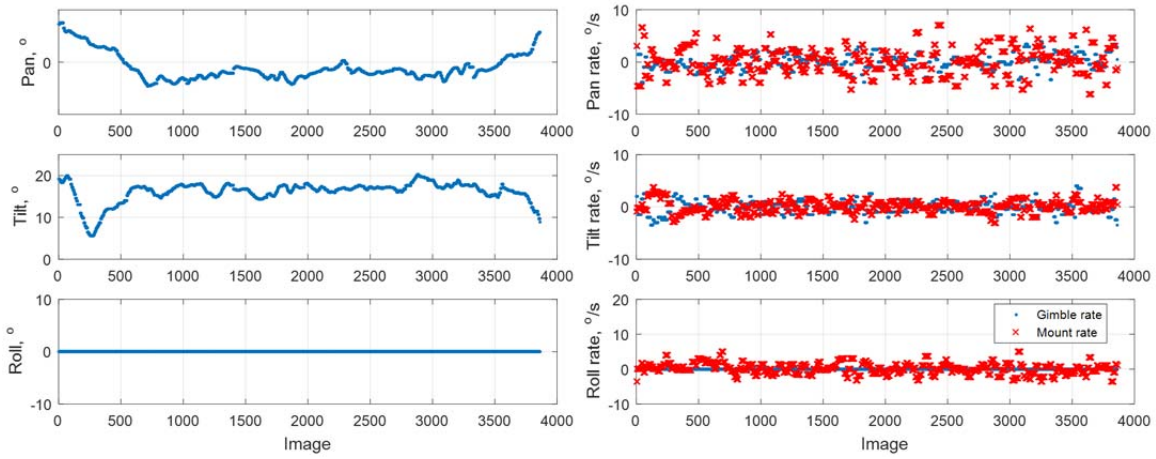


Figure 10. Roll, Tilt, and Pan (left) and Roll, Tilt, and Pan Rate (right) Flight Data

B. LITERATURE REVIEW

Research in the fields of control theory and computer vision has resulted in a number of methods recommended for real-time runway detection and tracking. This section reviews previous research that provides the foundation for this thesis. Based on these techniques and methodologies, this thesis seeks to improve detection and tracking performance and present alternatives to improve accuracy or processing time. Broadly, the relevant research areas include runway detection and tracking, pose and attitude estimation, and control theory.

In "Vision-based Runway Recognition for UAV Autonomous Landing," Jiajia Shang and Zhongke Shi (2007) describe one of the most complete methods for runway detection and implementation in an aircraft control system. Generally, Shang and Shi's methodology includes image preprocessing, runway area location, edge detection, and runway edge determination. The preprocessing stage uses a 3x3 crisscrossing median

filter, a standard and reliable image processing technique, to eliminate noise in the image (Shang and Shi 2007, 112). The runway location stage uses a double grayscale threshold technique that eliminates pixels below a predetermined threshold from an image, leaving only those regions in the image that lie on the runway (Shang and Shi 2007, 113). Grayscale thresholding reduces processing time because it only analyzes intensity information. While quicker, it also eliminates hue and saturation information that might contribute to the accuracy of detection. Shang and Shi chose to use the Sobel operator to detect vertical and horizontal lines in the image for further processing (2007, 113). The Sobel operator uses vector summations from a 3x3 neighborhood to gather gradient information for each pixel (Sobel 2014). Following edge detection, Shang and Shi used the Hough transform to find lines in the parameter space that correspond to the runway edges (2007, 114). This process is well suited for accurate real-time detection of runway edges and this thesis uses the general framework of this algorithm.

Other common runway detection methods involve matching real-time images to templates of a runway to determine distance and position. This method is typically more time intensive and prior images of the runway are required for successful detection and tracking. While this technique may be feasible for UAVs that consistently use the same runway for landings, it does not meet the requirement of robustness as previously discussed in this chapter.

C. THESIS ORGANIZATION

As previously stated, this thesis specifically focuses on the image processing component of the larger vision-based landing system, which aims to accurately detect and track a runway in real time. The systems engineering feasibility analysis starts with a complete description of the framework and methodology of the runway detection and tracking algorithm in Chapter III. This chapter outlines the pre-processing steps necessary to localize the runway and eliminate noise and clutter. Next, Chapter IV reviews the chosen edge detection technique, the Hough transformation to identify the runway edges within the image, and the error detection and correction methodology. Then, Chapter V outlines the result of the feasibility analysis and includes a discussion of the algorithm's

performance. This thesis ends with Chapter VI presenting conclusions from this research and outlines future work for autonomous landing system development.

THIS PAGE INTENTIONALLY LEFT BLANK

III. FRAMEWORK AND METHODOLOGY: PRE-PROCESSING

This chapter describes the detailed pre-processing steps of the runway detection algorithm. These steps include image thresholding, morphological operations, image masking, and image sharpening. The algorithm was developed and executed in MATLAB. Use of any predefined functions from the MATLAB image processing toolbox is explicitly stated.

A. THRESHOLDING DESCRIPTION AND PURPOSE

Thresholding is the process of segmenting an image based on a specific characteristic such as color or intensity (Cheng, Sun, and Wang 2011, 28). Thresholding is a fundamental image processing technique because it is a simple and computationally efficient way to focus on relevant areas while eliminating irrelevant regions. Thresholding results in the isolation or elimination of a specific region within the image for further processing. While grayscale intensity thresholding is a common technique, all color spaces are can be used for thresholding but determination of the correct color space requires trade-off analysis for accuracy and processing time.

The purpose of image thresholding for runway detection is twofold. First, it is the initial step for localizing the runway within the given frame. With properly defined thresholds, the algorithm should effectively highlight the area of interest, which is the runway. Additionally, by highlighting the relevant areas of an image, thresholding should save time in later stages of the algorithm by limiting the number of pixels processed through more complex and computationally intensive steps. Based on the results of this research, it is evident that thresholding often will play the greatest role in eliminating noise and clutter from the image. While thresholding has the potential to eliminate large unnecessary areas, it is still important to avoid elimination of important data within the image when constructing the thresholding limits.

1. Description of Color Spaces

Alvy Ray Smith introduced the HSV (hue, saturation, and value) color space in 1978. Smith (1978, 3) described hue as "the dimension with points on it normally called red, yellow, blue-green, etc.," saturation as "the departure of a hue from achromatic, i.e., from white or gray," and value as "the departure of a hue from black." Hue, saturation, and value can describe colors in a more intuitive fashion. Smith (1978, 4) likened the HSV color model to the process used by an artist to create his or her paint, where he or she would choose a base color and add white or black paint in order to achieve the desired saturation and value. Smith's primary objective in creating the HSV color space was to provide an alternative to the RGB (red, green, blue) color space, which was widely used but difficult to conceptualize. A visual representation of the conical HSV space is shown in Figure 11.

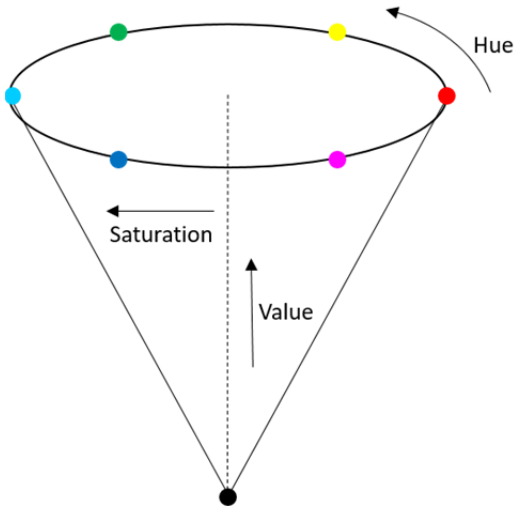


Figure 11. Conical Representation of the HSV Color Space

Electronic devices and monitors typically use the RGB color space to produce and display images. While there are variations of the RGB model, they all use some combination of red, green, and blue channels to create the full spectrum of colors. The chromaticity of a color defines this red, green, and blue combination and the mixing process is a close reproduction of how the receptors in the human eye process colors via wavelength combinations (Joblove and Greenberg 1978, 20–21).

Grayscale images eliminate all hue and saturation information, leaving intensity, or value, as the only remaining characteristic. All pixels within a grayscale image will therefore carry some value at or between the extremes of black and white. The simplicity and intuitive nature of grayscale images make them popular for image processing. Grayscale images also provide an advantage in processing speeds, as the images will only contain one channel for intensity compared to three channels in the HSV and RGB color space.

2. Comparison of Color Spaces

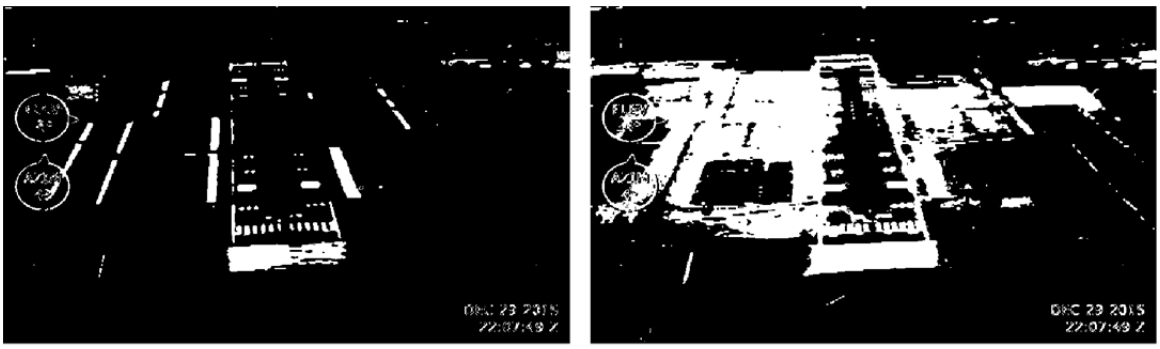
Determining the best color space for the thresholding process required a thorough comparison based on accuracy and computation time metrics. The three options were the RGB color space, the HSV color space, and intensity for grayscale images.

The RGB color space was eliminated early in the comparison process because runway surfaces tend to include little variation in chromaticity, making it difficult to achieve any degree of accurate thresholding using red, green, and blue channels. Red, green, and blue channel thresholding is most effective when the target or area of interest is a unique color and its particular combination of red, green, and blue allows for distinct separation from the surrounding environment. Unfortunately, this is not the case for the dark runway surfaces and white runway markings, which only fall at the extreme values of the red, green, and blue channels. If anything, RGB analysis demonstrated the importance of using multiple image characteristics for analysis rather than simply chromaticity or intensity.

After comprehensive comparison, research and experimentation favored the HSV color space over the grayscale color space. Implementation in the MATLAB environment proved that the grayscale thresholding process is actually quite involved. The RGB color space is the most common choice for image storage on electronic devices, so the algorithm must first convert the image into the grayscale color space from the RGB space. Following this conversion, each pixel in the image must undergo a gradient calculation in order to determine the maximum and minimum intensity values for the thresholding limits. It is possible to choose predefined threshold limits that do not use a

relative gradient relationship, thus eliminating the need for the gradient operation. However, this has a decidedly unacceptable effect on accuracy, as intensity changes considerably with distance from the runway, environmental conditions, and noise. Experimental results comparing HSV and grayscale thresholding times support the conclusion that HSV thresholding is actually the superior method. A test of 30 trials for each method determined that the difference between HSV and grayscale mean processing times was 0.147 seconds ± 0.010 seconds in favor of the HSV color space (see the Appendix for full results). Considering real-time requirements for the algorithm, 0.147 seconds is a significant difference.

The HSV color model was also the preferred choice in accuracy comparisons. The inclusion of hue and saturation data in the thresholding process resulted in a more robust controller that could accurately localize the runway region through a wider range of distances, varying environmental conditions, and image noise levels. When only working with intensity values, the user is limited to the manipulation of one channel for thresholding. Wide ranges will eliminate noise and clutter but may result in the elimination of relevant areas in the runway region. Narrow threshold ranges will include all the relevant areas of the runway but increase noise and clutter in the output image. The difference between these extremes is small but the results are drastically different, as shown in Figure 12.



Narrow intensity threshold from 0.65 to 0.85 (left) compared to a wider intensity threshold from 0.60 to 0.90 (right).

Figure 12. Comparison of Narrow and Wide Intensity Threshold Margins for a Grayscale Image

It is evident in Figure 12 that choosing threshold limits is a sensitive process. The intensity ranges only varied by a value of 0.05 for the upper and lower limits, but the results are dramatically different. Considering the uncontrollable factors that can affect image intensity, relying on a single image characteristic to achieve robustness in the thresholding process is challenging at best. In the HSV color space, the combination of hue, saturation, and value channels adds redundancy in the thresholding process, improving its resilience to variation due to uncontrollable factors. Therefore, HSV thresholding was the more accurate method in almost all trials.

3. Applied HSV Threshold Limits

The most appropriate approach to determining the upper and lower HSV threshold limits was through trial and error. The greatest challenge was finding limits that worked at both long and short distances from the runway. Extensive analysis of 3000 frames of approach images to Monterey Regional Airport and incremental adjustments to the upper and lower limits revealed the appropriate threshold range, as defined in Equation 1.2.

$$\begin{aligned} \text{Hue} &= (0 \le h \le 0.440) \cup (0.700 \le h \le 1) \\ \text{Sat} &= (0 \le s \le 0.200) \\ \text{Val} &= (0.637 \le v \le 1) \end{aligned} \tag{3.1}$$

The hue range includes all portions of the color wheel except those in the blue to cyan range. Excluding these hues is an efficient way to eliminate clutter and noise commonly resulting from the sky. Future applications in aircraft control may use the sky and the horizon line as aids to determine attitude and elevation, but these aspects are irrelevant to the runway localization and detection process.

The saturation limits range from 0.00 to 0.200 because runway markings are always white or light gray, which correspond to lower saturation values. The runway and taxi surfaces are made from concrete and asphalt, which will generate lower saturation values in images, thus aiding the localization process. This tends to create contrast with the surrounding environment, which usually includes saturated colors found in trees, grass, and other common airport environments.

The value limits favored brighter colors that highlight the white runway markings. For runways with black asphalt surfaces, the distinction between the markings and the runway surface is clear, making value thresholding even more useful. The distinction between markings and the surface is less clear on runways with lighter concrete surfaces, which slightly dilutes the effect of value thresholding.

Thresholding effectiveness is lowest at either extreme, both long and short, with performance at mid-range proving to be the most accurate. A comparison of long, medium, and short-range thresholding is illustrated in Figures 13 and14.

Figure 13. Original Approach Images at Long, Medium, and Short Range

Figure 14. HSV Threshold Images for Long, Medium, and Short Range

There is a large deviation in the results of the thresholding process. In some cases, thresholding will only capture the runway markings. In other cases, it will capture the entire runway as well as portions of the environment surrounding the runway. Both scenarios are acceptable as long as the thresholding process generally localizes the runway and eliminates large portions of unnecessary information from the image. Ideally, the resulting binary image will only include the runway markings on and around the runway, but an exact solution at all ranges and in all environments is impossible to

achieve. The next step, which applies morphological operations, will work to normalize the result of the thresholding process.

B. MORPHOLOGICAL OPERATIONS

This section provides a brief description of the purpose of morphological operations and an overview of the primary operations and functions used in the algorithm.

1. Background and Purpose

Morphology aids in the identification and analysis of shapes within an image (Dougherty and Lotufo 2003). The objective of image processing is often to identify or preserve some boundary, area, or region, which makes morphology invaluable to the field of computer vision. Some of the first uses of morphology date back to experimentation by Kirsch, Cahn, Ray, and Urban (1957), in which computer code completed several image pre-processing steps, removing humans from time-intensive operations. From those foundations, the process and purpose behind morphology largely remains the same. In the context of this research, the purpose of morphology is to further localize the runway area and capture the outline of the runway and any markings on the surface.

2. Erosion and Dilation

The two basic morphological operations are erosion and dilation. Traditionally, algorithms use these operations on binary images to identify shapes and regions more accurately. Working with binary images also decreases complexity as matrix operations only include ones and zeros.

Dilation is the process of enlarging a shape or region within an image by expanding the boundary of that shape (MathWorks 2017i). The number of pixels used to enlarge the region depends on the structuring element used in the dilation process (MathWorks 2017i). For MATLAB functions, the dilation rule states, “the value of the output pixel is the maximum value of all the pixels in the input pixel’s neighborhood” where the output pixel and neighborhood pixels are determined by the shape and size of the structuring element (MathWorks 2017i). Dilation can recapture portions of an image

that may have inadvertently been lost in pre-processing steps. An example of two shapes dilated by a 30x30 square structuring element is shown in Figure 15.

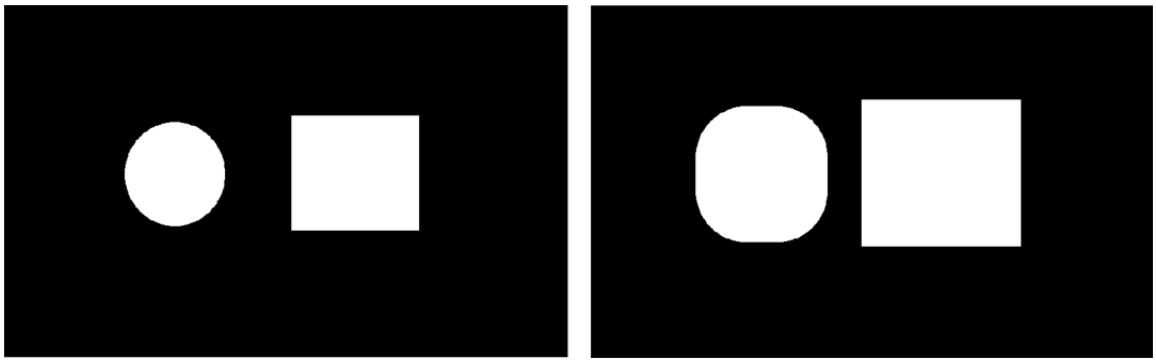


Figure 15. Example of Dilation Operation with 30x30 Square Structuring Element

Erosion is the contrasting morphological operation, where the boundary of a region is reduced, thus shrinking the area of the shape (MathWorks 2017i). In this scenario, the minimum value within a neighborhood of pixels defined by a structuring element determines the value of the output pixel (MathWorks 2017i). Erosion eliminates noise and unwanted clutter in an image. An example of the erosion operation with a 30x30 pixel square structuring element is shown in Figure 16.

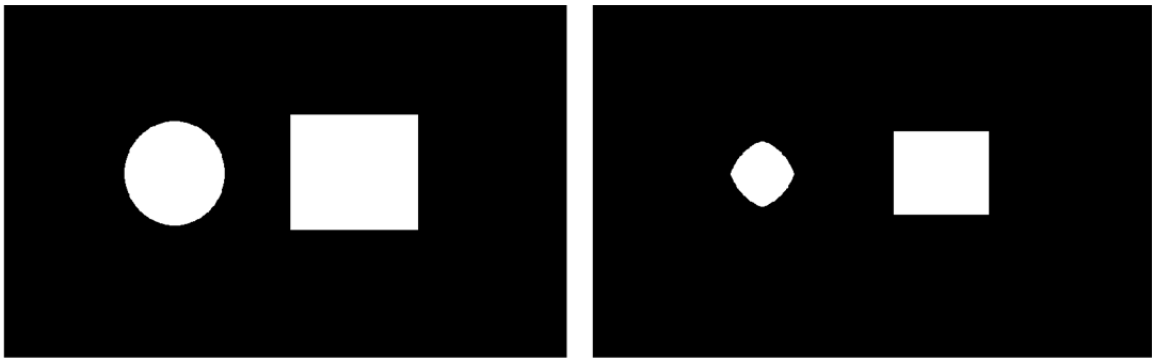


Figure 16. Example of Erosion Operation with 30x30 Square Structuring Element

3. Opening and Closing Operations

Opening and closing operations combine the erosion and dilation processes to improve the image while preserving the shape and boundary of the objects of interest. This is specifically useful when the objective is elimination of noise and clutter.

The opening operation is the sequential application of erosion followed by dilation. Erosion will eliminate smaller, unwanted binary regions and dilation will restore the region of interest to its original size. An example of clutter elimination is shown in Figure 17, with a square structuring element used to preserve the square shape of the region of interest.

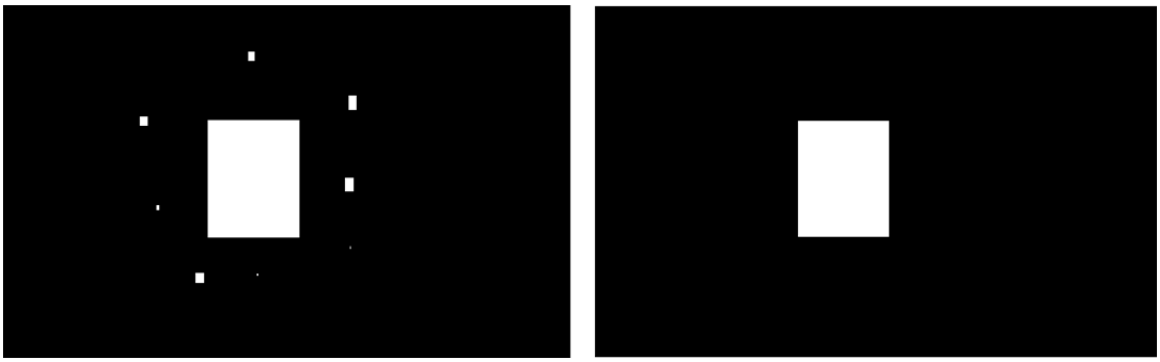


Figure 17. Opening of an Image with 20x20 Square Structuring Element to Eliminate Clutter

Closing involves the sequential application of dilation and erosion. The most common use for the closing operation is to remove gaps or holes in a region of interest. A simple example of this application is illustrated in Figure 18, where a 20x20 square structuring element fills gaps in a rectangular region.

Figure 18. Closing of an Image with 20x20 Square Structuring Element to Fill Gaps

4. Application in Runway Detection Algorithm

For demonstration purposes, a final approach image taken approximately 37 seconds from touchdown will demonstrate the result of morphology in the context of this algorithm. The original image and thresholded image are shown in Figure 19.

Figure 19. Original and Thresholded Image for Use in Morphological Operation Demonstration

The first operation was a filling operation available in MATLAB's image processing toolbox labeled "imfill" (MathWorks 2017a). The function acts as a closing operator, filling in gaps or spaces in enclosed regions within a binary image. The result of this step is illustrated in Figure 20.

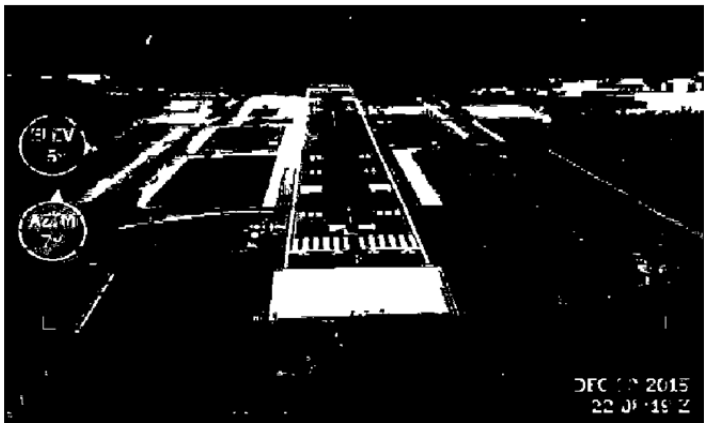


Figure 20. Image Produced after Executing “imfill” Function in MATLAB

With all holes filled in the binary regions of the image, the next step is to eliminate noise and clutter. An opening operation is used to eliminate the smaller clusters of pixels that passed through the threshold process but do not contribute to runway localization. MATLAB’s “bwareaopen” function, also included in the image processing toolbox, was used to accomplish this task (MathWorks 2017b). This function uses the opening operation to eliminate pixels below a defined threshold area. The runway detection algorithm uses an area threshold of 30 pixels for the opening operation, which was determined through a process of trial and error. Thirty pixels will eliminate most noise but retain all areas on or near the runway. The final product of the “bwareaopen” operation is shown in Figure 21.

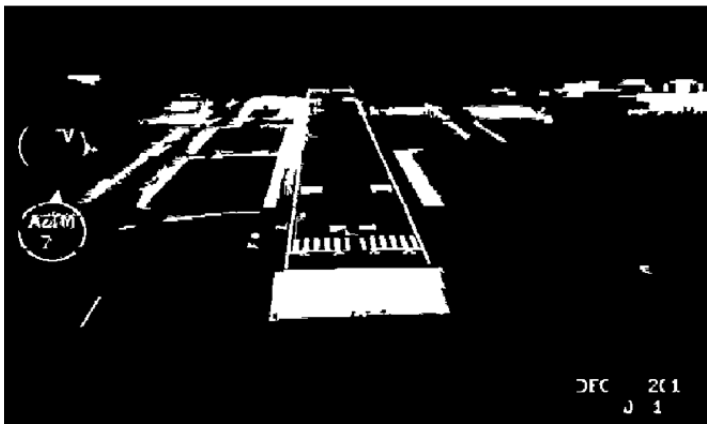


Figure 21. Image Produced after Executing “bwareaopen” Function in MATLAB

It is apparent in Figure 21 that some of the markings on the runway surface were lost in the morphology process. This is acceptable because the primary objective is to identify the runway edges and later stages of image processing can recapture lost features.

Dilation is the next morphological operation applied to the binary image. The objective of this final step is to recapture any area of the runway that were lost or eliminated in previous steps. The dilation operation uses a vertical, rectangular structuring element, as runway markings and components are typically vertically oriented rectangles. The result of this process is illustrated in Figure 22.

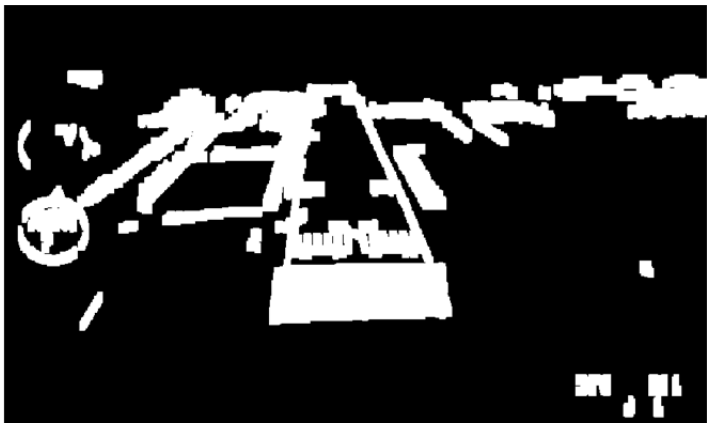


Figure 22. Image Produced after Executing Dilation with 10x4 Vertical Structuring Element in MATLAB

Ideally, dilation will capture the complete outline of the runway with no breaks or gaps, as shown in Figure 22. However, the algorithm will occasionally fail to capture every portion of the runway edge and surface. A robust design and redundancy throughout the algorithm will allow later steps to either correct the mistake or overcome the lack of data.

The final operation is a second iteration of the "imfill" function, which refines the morphology output (MathWorks 2017a). If the algorithm identifies the entire perimeter of the runway without gaps or breaks, the second filling operation will fill the interior

surface of the runway, an added benefit for the creation of an image mask. The result of the second iteration of the "imfill" function is shown in Figure 23 (MathWorks 2017a).

Figure 23. Image Produced After Executing Second Iteration of "imfill" Function in MATLAB

C. IMAGE MASKING AND FILTERING

This section covers the masking and filtering steps in the runway detection algorithm. Included are descriptions of each process, the purpose and justification for each step, and a brief overview of the types and sources of noise common to images.

1. Image Masking

Image masking is a process that captures a desired portion of an image while setting all other background pixels in the image to zero, or black. For the runway detection algorithm, the resulting binary image from the morphology process serves as the mask for the original runway image. Filtered sections of the original runway image will replace any white sections of the binary image that remained after thresholding and all morphological operations. In other words, image masking merges the binary image with the filtered version of the original image. This concept is best illustrated by Figure 24, in which the final masked image is displayed.

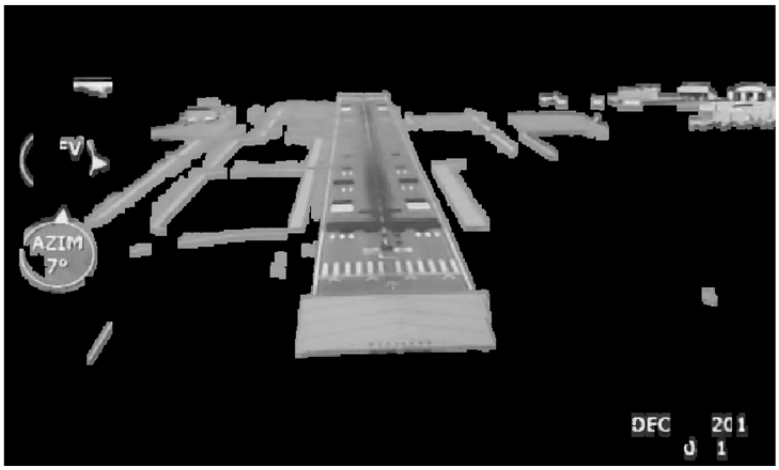


Figure 24. Image Mask of Binary Image and Median Filtered Runway Image

Aside from a few areas of clutter, the final product should include the runway region with as many markings visible on the surface as possible. An accurate image mask will lead to improved edge detection and line detection via the Hough transform at later stages.

Naturally, the algorithm will not always work perfectly and portions of the runway will be missing from the final image mask. However, this should result in failure. Even with partial data, the algorithm can achieve accurate detection of the runway edges. Designing a robust detection algorithm requires that varying degrees of accuracy must be acceptable at the image mask stage. Multiple examples of less accurate image masking results are shown in Figure 25. All of these images successfully resulted in accurate runway detection. At minimum, the image mask should at least include the edges of the runway, from start to finish, without breaks.

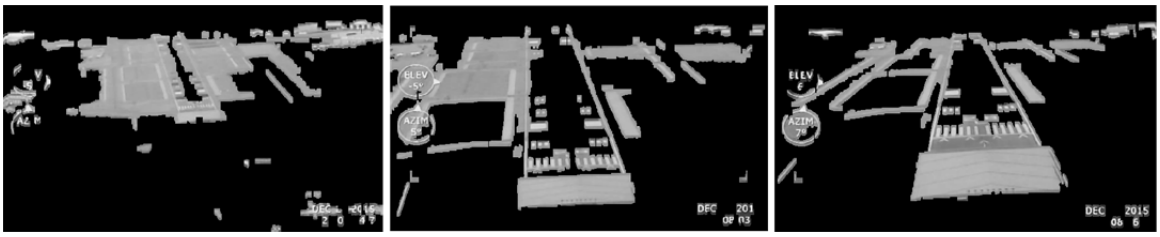


Figure 25. Examples of Various Levels of Image Mask Accuracy

2. Image Filtering

Image filtering is a pre-processing technique to eliminate noise in an image. There are a number of approaches to filtering, each effective at handling various levels and types of noise. Choosing an appropriate filter involves analyzing the type of noise affecting an image and determining acceptable computation time. Specifically, this research considered the effects of three types of noise: Gaussian, salt and pepper, and quantization noise.

Gaussian noise is random additive noise that follows a Gaussian distribution and affects all pixels in an image (Dangeti 2003, 6). Internal components and processes of the camera, such as amplification, typically introduce Gaussian noise (Patidar, Gupta, Srivastava, and Nagawat 2010, 46). Due to its pervasiveness, all image processing algorithms should account for Gaussian noise.

Salt and pepper noise is a series of intensity spikes that randomly affect an image (Dangeti 2003, 7). The affected pixels will assume either a maximum or a minimum value as a result of malfunctions in the sensors of the camera or errors in the analog-to-digital conversion process (Dangeti 2003, 7). The name is derived from the salt and pepper spikes that are evident in the final image as the noise results in either white or black pixels. Again, salt and pepper noise is a common issue in image processing and computer vision so efforts should be made to reduce its effects.

Quantization noise occurs during image compression, which is required for JPEG images. The final approach images used for this research were stored in JPEG format in order to store a vast quantity of images; therefore, quantization error was a concern. The conversion process compressed 8x8 blocks of pixels resulting in the loss of information and a subsequent blocking effect in the image (Quijas and Fuentes 2014, 1). The more the image is compressed, the more apparent the blocking effect (Quijas and Fuentes 2014, 1).

There are many image filtering techniques available and the techniques grow in number and complexity as the field of image processing advances. This research considered two filtering methods, the mean filter and the median filter. These time-tested and proven techniques provide solutions to a number of image processing issues and they

are relatively simple in application. Effectiveness and computation time were the evaluation factors for each filter.

The mean filter averages the intensity values of a predefined window of pixels and replaces the center pixel, the pixel of interest, with that average value (Dangeti 2003, 12). This results in a smoothing or blurring effect across the image that eliminates random noise. The result depends on the severity of the noise as well as the size of the chosen filter. Larger filters will result in more smoothing or blurring (Dangeti 2003, 12).

Median filtering works in much the same way but it is a nonlinear process (Dangeti 2003, 18). The median filter also uses a predetermined window size to evaluate neighboring pixels, but the median value from the neighborhood replaces to center pixel of interest (Dangeti 2003, 18). This also has a blurring effect on the image. The advantages of the median filter are twofold. First, it is more robust in the sense that it eliminates outliers, as is true of any process that uses medians instead of means (Dangeti 2003, 19). Many also argue that the median filter is better suited for edge preservation because it must use the value of an actual pixel from the neighborhood (Dangeti 2003, 19). As there is not much difference between computation time for the mean and median filter, these advantages make it the better choice for application in the runway detection algorithm. Runway edge detection is the primary goal, so edge preservation should be a valued characteristic for a filter.

To reduce processing time, median filtering should apply only to the image mask, so only select regions of the image require computation. However, selective application of the filter is difficult and time intensive in MATLAB, so the algorithm filters the full original image prior to masking for proof of concept. The result of applying the "medfilt2" MATLAB function for a 3x3 neighborhood is shown in Figure 26 (MathWorks 2017c). The filter reduced noise levels within the image at limited expense to the runway edges and markings. An unfortunate consequence of median filtering is that the algorithm must first convert the original image to a grayscale image because it only filters pixel intensity values, thus adding additional computation time. Again, real-time application should only apply median filtering to the required areas of the image mask; this section only models the result of filtering due to constraints in MATLAB.

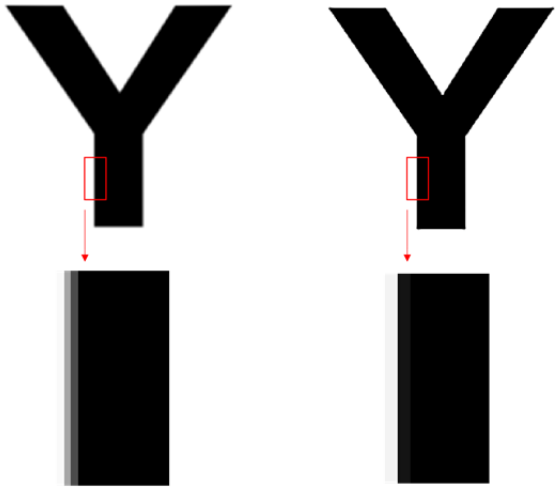
Figure 26. Intensity Comparison for Unfiltered and Filtered Runway Images

D. IMAGE SHARPENING

Image sharpening is yet another fundamental operation in image processing. This section provides a brief overview and background of the process, its purpose in the context of edge detection, and its formal application in the algorithm.

1. Background, Description, and Purpose

Image sharpening is a technique used to improve detail within an image, particularly textures and edges. Unsharp masking, a common sharpening technique, intentionally blurs the original image, usually through Gaussian blurring, and subtracts the blurred copy from the original image (Cambridge in Colour 2017). MATLAB sharpening functions allow the user to control the radius of the Gaussian filter as well as the threshold for determining edges. A larger radius will affect a larger area around edges and higher thresholds will only apply the filtering effect to stronger lines within the image (MathWorks 2017d). If the unsharp masking process identifies an edge that meets the threshold, the filter sharpens the edge by an amount defined by the user. This "amount" setting affects the acuteness of edges, which is a measure of edge transition from dark to light (Cambridge in Colour 2017). Higher acuteness leads to a sharper transition, hence the name of the operation. An application of unsharp masking is demonstrated in Figure 27, where an image was intentionally blurred with a Gaussian filter and sharpened using MATLAB's "imsharpen" function from the image processing toolbox (MathWorks 2017d).



Gaussian blurred image (top left), sharpened image (top right), magnified view of blurred image edge (bottom left), and magnified view of sharpened image edge (bottom left).

Figure 27. Demonstration of Unsharp Masking Using a Gaussian Blurred Image

The unsharp masking process uses the intensity values of an image, but the output image can be in both color and grayscale. The function will simply convert the image between color spaces to meet the requirement. Figure 27 shows an ideal image sharpening process, with an edge between a perfect white background and a perfect black object. Real application of unsharp masking will almost never meet these ideal conditions but the process is still effective. For the runway detection algorithm, image sharpening is necessary because edge preservation is crucial for final application of the Hough transform. Without clear, defined edges, the edge detection process and resulting Hough transform calculation will not be accurate. The filtering process, while necessary, introduces additional blur to the image that sharpening can reduce. Although median filtering preserves edges to some extent, image sharpening refines all the major edges within the image and prepares the processed image for the final stages of runway recognition and detection.

2. Application in Algorithm

The objective of image sharpening in the context of the detection algorithm is to increase edge sharpness for lines on the runway while limiting the sharpening effect for

all other irrelevant areas. Variation of the radius, threshold, and amount inputs will achieve this objective. Ideal sharpening will only affect a small area around the edge of interest, so a smaller radius is ideal. The default radius of 1 pixel was used because it is an agreed upon standard and experimentation with lower and higher values yielded no improvement. Low thresholds will capture unnecessary details and textures within an image but excessively high thresholds will not effectively sharpen any edges. The thresholding and morphology steps executed prior to sharpening have eliminated most of the unnecessary regions in the image. For this reason, the algorithm uses a threshold limit of zero, accepting introduction of some noise as a result. The amount must be sufficient to highlight the edges within an image but not so high that the overshoot between the light and dark transitions of the line becomes unnatural or counterproductive to detection (Cambridge in Colour 2017). Through trial and error, the amount argument was set to 5, which significantly increased the sharpness of the runway edges without unnatural distortion in the image. The final product is shown in Figure 28.

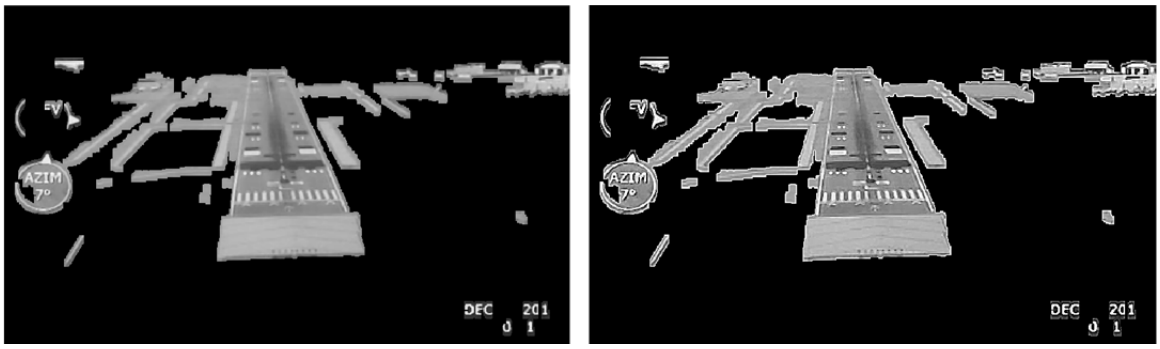


Figure 28. Comparison of Unsharpened and Sharpened Runway Image

THIS PAGE INTENTIONALLY LEFT BLANK

IV. FRAMEWORK AND METHODOLOGY: LINE EXTRACTION AND ERROR CORRECTION

This chapter describes the edge detection methods and line extraction functions specifically chosen to identify the runway edges within an image. It includes a complete analysis and comparison of relevant edge detection methods as well as a detailed overview of the Hough transformation process.

A. EDGE DETECTION

Edge detection is the final step before applying the Hough transform and calculating the position of the runway edges. This section provides a brief overview of edge detection, its purpose, and the various techniques used for edge detection. Also included are a comparison of methods and the justification for use of the Sobel operator.

1. Overview and Purpose

In a study and comparison of various edge detection techniques, Maini and Aggarwal (2009, 1) describe edge detection as "the process of identifying and locating sharp discontinuities in an image." These discontinuities are the points at which the intensity of an image changes, indicating the start or end of an edge (Maini and Aggarwal 2009, 1). The general approach to edge detection involves convolving a square operator with an image that detects these discontinuities and returns a binary image displaying all detected edges (Maini and Aggarwal 2009, 1). Gradient operators work by detecting the maximum and minimum values of the derivative of intensity values while Laplacian detection uses the second derivative to detect edges (Maini and Aggarwal 2009, 2). Laplacian methods are inherently more computationally intensive and adversely affect real-time detection, so this research ignores Laplacian methods in favor of gradient operators.

2. Comparison of Edge Detection Techniques

With the elimination of Laplacian methods, the two most popular gradient methods, the Sobel operator and the Canny algorithm, remain. The Sobel method

convolves a 3x3 gradient operator, shown in Figure 29, with the original image (Sobel 2014). The gradient operator sums the gradient values of orthogonal vectors resulting in a magnitude and direction value for a given neighborhood (Sobel 2014). The user can therefore detect lines along a specific gradient, usually broadly categorized as horizontal or vertical, as well as lines of a certain magnitude. At the time of its inception, the Sobel operator was notable for its performance with other available edge detection techniques and its superior computation time. Developments in edge detection over the years have led to its new label as a traditional and simple edge detection technique, but the Sobel method still merits widespread attention and application in the image processing field. More involved than the Sobel method, the Canny method is actually a combination of a several processes. It vastly increases accuracy while maintaining an acceptable level of complexity. The algorithm achieves three objectives as outlined by Canny: low-error rate, accurate localization, and a single response to each edge (Maini and Aggarwal 2009, 6). The general process for the Canny method is as follows: application of a Gaussian filter, convolution with a gradient operator, calculation of gradient magnitude and direction, line thinning via non-maximum suppression, and hysteresis to eliminate any gaps in the edge (Maini and Aggarwal 2009, 6–8).

-1	0	+1
-2	0	+2
-1	0	+1

+1	+2	+1
0	0	0
-1	-2	-1

Figure 29. Vertical Sobel Operator (left) and Horizontal Sobel Operator (right)

The Sobel and Canny methods share many of the same advantages and disadvantages. They are popular because they are simple to use and they can each capture edge direction and magnitudes. However, both are somewhat inaccurate and less robust than more complex edge detection techniques. When comparing the two methods, the Sobel operator is more computationally efficient, whereas the Canny method is more

accurate. Determination of the appropriate method depends on the trade-offs between advantages and disadvantages and the role and scope of edge detection in the greater system. The Sobel operator was the favorable method in this research for its simplicity and computational efficiency. Many of the steps taken in the Canny method were unnecessary due to previous image processing steps. For example, the Gaussian filter applied by the Canny method would be redundant and it would undo many of the favorable effects of the sharpening process. The difference between the two methods is best illustrated by their outputs, which are displayed in Figure 30. Each method was implemented in MATLAB with a vertical filter and threshold set to 0.015.

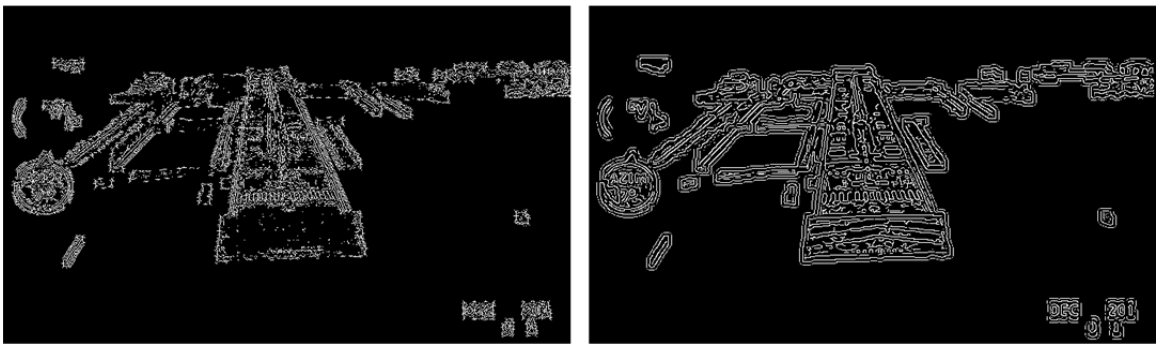


Figure 30. Comparison of Sobel Edge Detection (left) and Canny Edge Detection (right)

3. Application in Algorithm

The algorithm used Sobel edge detection via MATLAB's "edge" function from the Image Processing toolbox. This function allows the user to specify the edge detection technique as well as a series of input arguments including threshold, direction, and thinning (MathWorks 2017e). The threshold for the Sobel operator was set to a magnitude of 0.015. Through trial and error, the author determined that this value performed best at all ranges. The direction was set to vertical, which executed convolution with the vertical Sobel operator. For final approach images, the runway direction should align with the vertical axis of the frame. Resulting binary images from a variety of ranges are shown in Figure 31.

Figure 31. Sobel Edge Detection on Runway Images at Far, Medium, and Close Ranges, Respectively

B. HOUGH TRANSFORM

This section provides a brief background to the invention and development of the Hough transform, ultimately leading to its importance in the modern field of computer vision. Also included is an explanation of the transformation process and its application in the runway detection algorithm.

1. Background

Paul Hough first introduced the concept of the Hough transform as a means to detect patterns created by subatomic particles in a bubble chamber (Hough 1962, 3). The motivation behind the Hough transform was to reduce the amount of time required to analyze photographs of the particle by allowing machines to analyze the pictures and find patterns (Hough 1962, 3). This motivation makes this process well suited for analyzing runway approach images in real-time. Hough's patent describes a plan to transform points in a Cartesian plane into the slope-intercept parameter space in which intersection points would correspond to lines in the Cartesian plane (Hough 1962, 3). If the objective of image analysis is line or pattern detection, the Hough space makes the process much less time intensive. Patterns of interest become points of intersection in the Hough space, reducing what would normally be a complex and time-intensive search of the Cartesian plane.

Richard Duda and Peter Hart (1972, 2) improved the Hough transform by using polar coordinates to describe points in the parameter space via Equation 1.3.

$$\rho = x\cos\theta + y\sin\theta \tag{4.1}$$

This eliminates the possibility of undefined or infinite slopes in the slope-intercept plane (Duda and Hart 1972, 1). The transformation to the polar parameter space is illustrated in Figure 32. Hart and Duda (1972, 4) were also able to establish the following rules for the transformation: all points in an image frame correspond to a sinusoid, points in the parameter space are straight lines in the image, and points on the same line in the image frame will share a common point in the parameter space. This improvement to the Hough transform makes it extremely applicable in image processing and computer vision. Undefined lines in the parameter space become a non-issue and image processing algorithms need only to find peaks in the Hough space to identify lines in the image plane. The more sinusoids intersecting at a single point, the stronger the peak, and the longer and more defined the edge or line. Most computer vision algorithms use Hart and Duda's adaptation of the Hough transform.

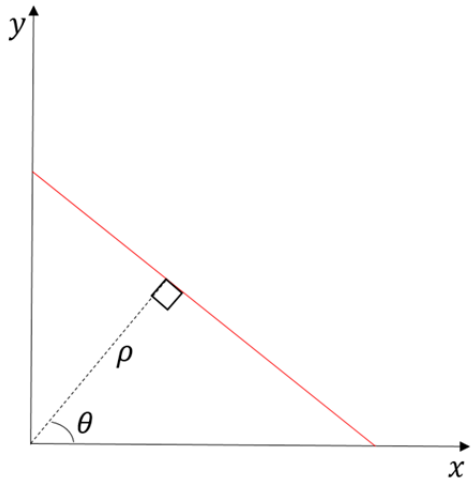


Figure 32. Transformation from x-y Image Plane to ρ-θ Parameter Space

2. Application in Algorithm

The Hough transform is the optimal process for identifying runway edges in an image. Considering its objective, the Hough transform is relatively efficient and it offers value in a real-time detection environment. The key to further reducing computation time is to limit the number of edges detected by the Sobel operator. Any clutter or excess lines

will only degrade accuracy and add to computation time. This is why the thresholding, morphology, and sharpening steps are so crucial to improving accuracy and efficiency.

The Sobel edge detected image serves as the input to the Hough transform process. As with most processing steps, binary images with clear delineations between edges and limited noise provide accurate results. The "hough" command in MATLAB will perform the Hough transform, using the binary image as the input and producing a matrix that describes the Hough space with rho and theta values (MathWorks 2017f). For added accuracy and error reduction, the search of the image plane and the resulting Hough space is limited to theta values between -35 and 35 degrees. This means that the lines the Hough transform seeks to detect are vertical lines, with extra room to allow for variations in aircraft position relative to the runway in the final approach stage. The parameter space and the associated binary input image are shown in Figure 33. Each sinusoid corresponds to a pixel in the binary image and all possible lines passing through that image from -35 degrees to 35 degrees.

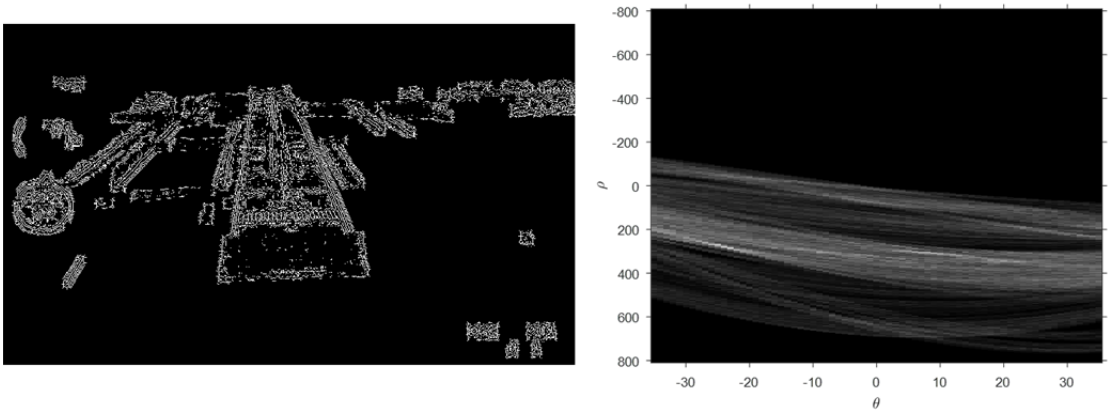


Figure 33. Binary Input Image and Rho/Theta Hough Parameter Space

A cursory glance at the Hough space will indicate a few likely intersection peaks corresponding to lines in the image plane. Points of specific interest are located near the negative 25-degree mark and the 25-degree mark in Figure 33. For the runway edges, two responses of similar magnitude but opposite angles are indications of an accurate match. If the aircraft is lined up with the center of the runway, the angles made by both edges of

the runway will be the same magnitude. In fact, these runway angles can aid in the estimation of aircraft roll positions. These peak intersection points are highlighted by yellow boxes in Figure 34 by using the "houghpeaks" command in MATLAB (MathWorks 2017g). This command is useful because it allows the algorithm to focus on a predetermined number of peaks. The two peaks with the largest number of intersections are most likely the runway edges, but this may not always be the case. For this reason, it is a good practice to analyze the top 5–7 peaks in the image, depending on the amount of noise and clutter.

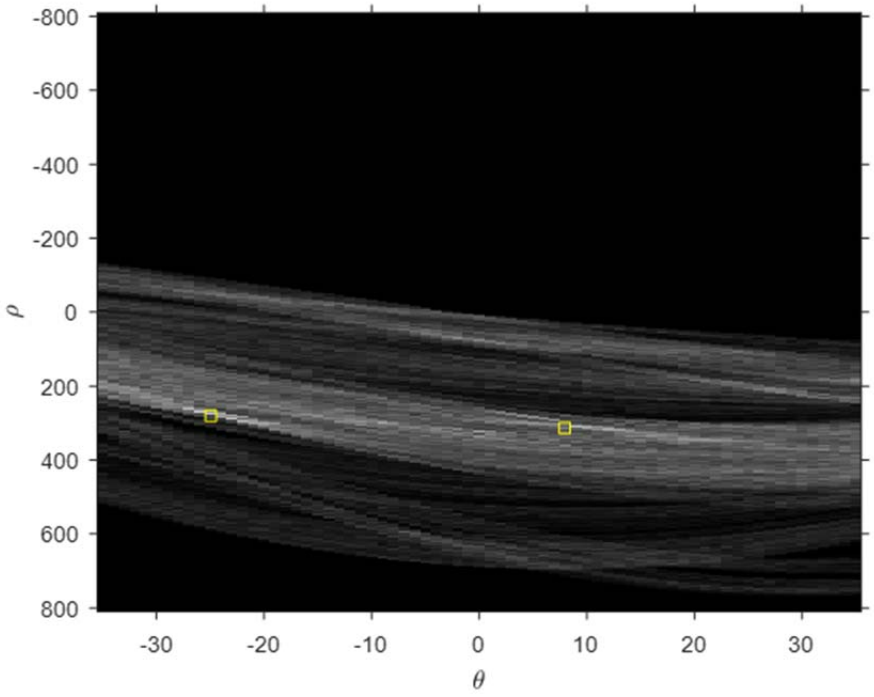


Figure 34. Hough Space with Top Peaks Highlighted

For the example shown in Figure 34, the top two peaks do correspond to the runway edges, but to prove this, the algorithm must convert the points back to lines in the image frame. To do this, the MATLAB function "houghlines" uses the peaks returned from the "houghpeaks" function to extract the appropriate rho and theta values that the function then into x–y plane values (MathWorks 2017h). Overlaying the resulting Hough

lines on the original runway image proves that the lines do accurately correspond to the runway edges, as shown in Figure 35.

Figure 35. Runway Image with Hough-Detected Lines Overlaid

Using a binary image collected via the Sobel method, the Hough transform can accurately detect the edges of a runway. The remaining challenge lies in correcting error and implementing the visual feedback in a control system.

C. ERROR DETECTION AND ADJUSTMENT

Error adjustment is critical to the success of the runway detection algorithm. If the algorithm is to be used in a real-time environment, it must achieve a high degree of accuracy. This section describes the process used to detect and correct errors within the algorithm.

1. Detection Issues Using the Hough Transform

The algorithm, as described in the previous sections, is robust; however, it will not provide total accuracy in all final approach scenarios. Attempting to pinpoint runway edges from a camera attached to a vibrating, high-speed vehicle is an inherently complex process. Issues are plentiful and experimentation with thousands of frames of final approach images exposed some of these problem areas.

As previously stated, the accuracy and speed of the Hough transform depends on the accuracy of the preceding steps of the algorithm. If thresholding and morphology can remove clutter and capture the entirety of the runway, then the Hough transformation becomes much simpler and more accurate. However, often the environmental conditions and image noise levels make detection much more difficult and simple operations such as thresholding, morphological opening, and closing are not suited nor were they designed to handle such complex scenarios. Therefore, noise and clutter in the input binary image are inevitable. Distance and environmental conditions that negatively affect visual range, such as clouds or fog, are common sources of these issues.

Even in ideal conditions, the Hough transform can provide false outputs. Combinations of noise and pixel alignment may result in peaks in the Hough transform that do not correspond to any edge or runway line at all. This example is illustrated in Figure 36. Clutter and noise in the center of the runway caused the algorithm to fail to identify the right runway edge. For unavoidable situations like this, added error detection and correction is required. The challenge lies in reproducing human-like error corrections in an automated process.

Figure 36. Inaccurate Runway Edge Detection via the Hough Transform Method

2. False Runway Edge Detection Adjustment

The algorithm tests for a series of conditions to ensure that the identified edges are correct. The first condition tests the proximity of the runway endpoints for both the right and left edges. Although the distance between these points will vary depending on

image resolution and distance from the runway, the algorithm defines a “no closer than” distance that indicates one or both of the edges are incorrectly identified. This “no closer than” distance is 20 pixels or less in the positive or negative x-direction. This distance represents half the distance between the runway edges in the image taken at the farthest point in the final approach of the aircraft. Therefore, there is no scenario in which the distance between runway edges should be any smaller. However, this value will also depend on camera properties such as resolution and stored image size so variation based on equipment and operating conditions is acceptable.

The second error scenario tests for distances between the runway endpoints that are too large. The algorithm considers any distance between the far runway endpoints greater than 150 pixels in the x-direction or 40 pixels in the y-direction as too large. The largest distance between the far endpoints in any frame on the final approach data was 100 pixels, so a value of 150 is provides a reasonable buffer. The far runway endpoints are the best choice for distance evaluation because there is less fluctuation in their separation compared to the closer runway endpoints. The separation between the runway start points can vary from 50 pixels at long distances to nearly 350 pixels near touchdown. The far runway endpoints simply provide a narrower range and reasonable values for error testing.

The algorithm begins by assuming that the top two peaks in the Hough space represent the runway edges. If any of the previous error conditions are satisfied, the algorithm revisits the Hough space and analyzes the top six Hough peaks to find the correct runway lines. The system then compares the top Hough peak to the remaining four peaks and tests the runway endpoints for the same error scenarios. When the algorithm identifies two lines that meet the given conditions, these become the identified runway edges for the frame. If no matches are found, the algorithm simply chooses the top two peaks and accepts the error, assuming that later error correction steps will limit the impact. The system does not conduct a more extensive search of the Hough space because it would result in unacceptable increases in computation time. Unfortunately, this means that the algorithm might have to accept a certain level of error in the process.

An added layer of error correction comes in the form of averaging. As the algorithm detects the runway endpoints, it stores the values in a matrix that holds the previous 15 values. When enough points are stored in the matrix, the algorithm averages the current endpoint positions with the previous values. This limits the effect of any errors that made it through the previous endpoint-checking stage. It also smooths the data to limit small disturbances. It is important to remember that the camera collecting the data is attached to the wing of the aircraft, which is exposed to mechanical vibrations and turbulence from the environment. This inherent shaking is visible in sequential viewing of the approach frames. The motion of the aircraft results in noticeable changes in endpoint position. Smoothing the data will be essential in a control system environment. Averaging acts as a low pass-filter, which eliminates small fluctuations that will negatively affect the performance of the aircraft control surfaces. If the control system receives constantly fluctuating position data from the runway detection algorithm, the control surfaces in the aircraft will also constantly fluctuate. Depending on the aircraft characteristics and flight dynamics, this fluctuation can reasonably result in a failure to land. Therefore, averaging frames is a good practice for future applications of the algorithm. Additionally, the camera captures images at 24 frames per second, so the algorithm is averaging the data in intervals shorter than one second. This should still provide a quick enough update rate to suit the control surfaces of the aircraft.

The matrix used to store runway data also allows the system to compare corresponding endpoint positions between frames. In other words, if the algorithm identifies a runway edge in which any of the points significantly differs from that of the previous frame, the system will detect an error. Specifically, if any of the endpoints falls outside a radius of 10 pixels from the previous endpoint, the algorithm detects an error and replaces the current endpoint values with those of the previous frame. To ensure the system is not caught in a loop of ever-increasing errors, the matrix will reset if the system detects more than seven errors in a row.

With all of these error-reducing measures combined, the accuracy of the system improves significantly. Depending on the nature of the operational conditions, the aircraft, and the camera, the details of the error detection conditions may vary. However,

the framework can be applied universally. The error correction capabilities of the algorithm are illustrated in Figure 37, where uncorrected and corrected versions of the same frame are shown side by side.

Figure 37. Uncorrected (left) and Corrected (right) Versions of the Same Runway Approach Frame

V. RESULTS AND ANALYSIS

This chapter reviews the performance of the runway detection algorithm based on accuracy and processing time. It will also provide analysis on which operations had the greatest effect on the overall performance of the algorithm.

A. ACCURACY

The algorithm processed 3000 images from a final approach in a Cessna 206 to Monterey Regional Airport. In total, this data covers approximately one minute and 40 seconds of flight time. The first and final frames are shown in Figure 38 for visual reference.

Figure 38. First (left) and Last (right) Images Used for Algorithm Evaluation

This research defines accurate runway detection as identification within a 10-pixel radius of the true starting or ending points of the runway. For an image taken at medium range, a 10-pixel shift in the position of both the start and end points could change the estimated runway angle by three degrees and the estimated runway length by 10 percent. These are the maximum levels of acceptable variation that could still reasonable lead to safe landing conditions. Additionally, the runway edge must include no breaks or gaps along the length of the runway.

In a test of all 3000 frames, the algorithm incorrectly identified the runway in 114 frames, which corresponds to an accuracy of 96.2 percent. Four additional tests

ensured the accuracy of the results. This test also utilized the error correction techniques of the algorithm. The accuracy of the detection algorithm without the error correction techniques was 75.9 percent. 722 frames of the original 3000 frames did not meet the accuracy standards, most of which were frames near the end of the final approach. This indicates the relative importance of the error correction process. It also indicates the negative effect that low altitude has on the detection algorithm. At lower altitudes, the runway consumes more of the image and its shape is comparatively more distorted than those at higher altitudes.

Frames that led to inaccurate detection of the runway had a few commonalities, the first of which is being taken from either extremely long or extremely short distances. The majority of the errors came from frames at the beginning or end of the runway data, which means that long or short distances and high or low altitudes degrade the performance of the image processing algorithm. The altitude ranges along the flight path at which accuracy decreases is shown in Figure 39.

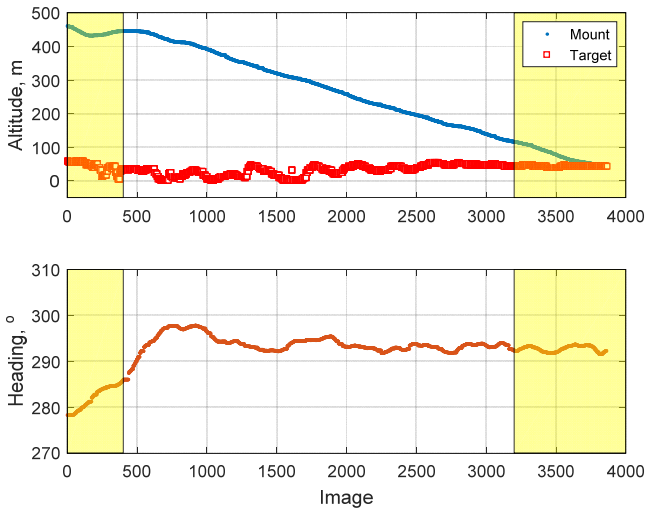


Figure 39. Accuracy Error Related to Altitude and Heading

At long distances, the most likely source of error is noise and environmental interference. Even in near-perfect visibility conditions, the clearness of the image will only improve as the distance becomes shorter. For the algorithm, this means that edges

are less delineated and sections of the runway can blend in with the surrounding tarmac of the airport. At shorter distances, the aircraft is closer to the ground and the view of the runway is comparatively distorted. Generally, these frames only include the concrete and asphalt surfaces of the runway and the horizon line. Saturation and value characteristics become increasingly more important at these distances as a result. Losing differentiation in hue will undoubtedly result in a loss of accuracy. It is also apparent that the runway endpoints approach the vanishing point in the image at shorter ranges. As a result, the endpoints become even more difficult to identify. The image frames that correspond to the start of processing degradation are shown in Figure 40.

Figure 40. Maximum (left) and Minimum (right) Distance for Image Processing Degradation

Clutter was also a common factor in incorrectly identified frames. Often larger and brighter elements near the runway, such as buildings, would pass through the filtering and morphology steps. If these elements happened to align with one of the runway edges and they were close enough to the runway surface, the algorithm often identified the clutter as continuations of the runway edge. Normally, the error correction process will identify these errors, but sometimes there are no alternative peaks in the Hough space to replace the false peaks.

There is room for improvement in accuracy, and the level of accuracy will depend on the application of the algorithm. If the system is intended for use on smaller, easily controllable aircrafts and large runways, accuracy is not as much of an issue. Application

in an aircraft carrier environment would require nearly perfect accuracy. These are extensions of the system in which the operator may need to make adjustments to achieve increased accuracy, but this research proved that the basic framework can operate at a successful level.

B. PROCESSING TIME

MATLAB scripts and functions are not always conducive for evaluation of real-time application. An aircraft would never use a MATLAB environment to implement real-time computer vision and aircraft control. MATLAB functions often call upon various toolboxes and sub-functions that require additional loading times. Sometimes MATLAB loads and implements subroutines in C or C++ to execute a function, as is the case for the filtering operations. However, it is much easier to demonstrate the framework and methodology in MATLAB. MATLAB can still offer insight and broad implications for real-time evaluation. Therefore, this discussion will only offer recommendations for reducing processing time based on MATLAB execution.

Data collected from five approach frames processed five times each revealed patterns as to which functions and operations consume the most processing time. The images vary in their distance from the runway to avoid bias due to distance. The "imfill" function, median filtering, and image sharpening were consistently the top three time-consuming processes. These are all matrix operations or convolution processes that must calculate values for every pixel in the image so it is not surprising that they are the most computationally inefficient. On average, the filling, sharpening, and filtering operations consume 11.2 percent, 10.5 percent, and 10.2 percent of the total processing time, respectively.

A possible explanation for the computation time of the "imfill" function is that it is not a true closing operation. It is a hybrid function that uses a more complex technique to fill in holes in binary regions of the image (MathWorks 2017a). The algorithm also uses two iterations of the "imfill" function, further explaining its increased processing time. While the computation is time consuming, it is still worthwhile to include both iterations of the morphological operation.

Image filtering is difficult to avoid, but limiting the area that requires filtering is a good practice for cutting computation time. The equipment used is also vital for the determination of the level of noise the algorithm must handle. Some cameras can provide clearer images with less noise, but they will come at a cost. Strategic placement of the filtering process in the algorithm will ultimately determine its efficiency. If implemented after all morphological operations and masking, filtering will be much more computationally efficient.

Image sharpening is the only process that could reasonably be removed from the algorithm, but only if conditions allow. Again, this is largely a factor of the operating environment, specifically the visual conditions, and the level of noise due to camera operations. With these factors reduced below an acceptable level, the algorithm can reasonably provide accurate results without the sharpening process. However, achieving those conditions is not easy, and it will likely be expensive. Efficient use of the sharpening threshold and amount settings can further reduce the computation time if sharpening is still necessary.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS

This chapter offers a comprehensive review of the research objective and focus, the framework and methodology of the runway detection algorithm, and results. It also offers lessons learned from the process and options for future work.

A. REVIEW OF RESEARCH QUESTION

The objective of this research was to conduct a feasibility study for the application of a runway detection algorithm for use in a vision-based autonomous landing system via a systems engineering approach. The system should be evaluated on its ability to detect the edges of a runway and any other relevant runway markings in an accurate, robust, and cost efficient manner. On a broader scale, this research assesses the feasibility and application of vision-based control in a broad array of autonomous systems. However, unmanned aerial vehicles propose a unique challenge to developing robust and autonomous decision-making systems.

B. CONCLUSIONS

Computer vision can accurately detect and track a runway in the final approach stages of flight. This research outlined a framework and methodology that proved its worth with an accuracy of 96.2 percent in final approach testing. Additionally, the framework supports real-time implementation in an aircraft control system.

Applying computer vision techniques to runway detection and unmanned vehicles requires a system that is accurate, robust, and computationally efficient. An accurate algorithm will utilize a multi-stage approach that constantly works to localize the runway region and eliminate noise and clutter. The algorithm should also utilize error detection and correction techniques to avoid false positives or misidentification of runway edges. These layers of image processing steps add an aspect of redundancy to the algorithm and improve overall performance.

A robust algorithm will be able to work at all distances expected in the operating environment, through various levels of noise, and in a degraded visual environment. It is

easy to construct an algorithm that will detect a runway in a very limited set of circumstances. The challenge lies in designing a system unaffected by external factors. Hue, Saturation, and value thresholding is a convenient way to increase robustness because it allows the algorithm to use more than just the intensity characteristic to localize the runway at limited computational expense. The more data available to the system, the more accurate it can be. Median filtering is also an efficient way to reduce Gaussian, salt and pepper, and quantization noise that will undoubtedly be present in an image. Combining these strategies will increase the operating range of the system.

Lastly, a computationally efficient algorithm will accurately detect the runway edges in the minimum amount of time, ideally for implementation in real-time. If the algorithm limits the amount of noise and clutter in an image, it will reduce the computation time for operations that require convolution and other complicated procedures like the Hough transform. It is acceptable to sacrifice accuracy in favor of computation time if the results are still reasonably in the desired range, as is the case with using the Sobel operator instead of the more accurate Canny method. This framework strikes a delicate balance between efficiency and accuracy.

C. FUTURE WORK

Future work would primarily include implementation of pose and attitude estimation as well as the development of a full aircraft control system with visual feedback as outlined in Chapter I. By doing so, the feasibility study could be extended to the complete vision-based landing system.

While this thesis conducted a thorough description and analysis of the image processing system within the constraints of the available data, the system still requires additional testing. Specifically, future work should collect data from a wide variety of runways and geographic locations to test the effects of background and clutter on the accuracy of the algorithm. Additional testing should also cover various light levels, degraded environmental conditions, different types of aircraft, varying speed, and varying glide slopes. If the landing system is to be truly robust, it must successfully operate in all of these conditions.

While the results of the thesis have proven that a vision-based landing system is at least partially feasible, they also indicate that the runway detection algorithm struggles at extremely long and short ranges. If this vision-based approach is to be pursued, future work must focus on achieving accurate results at both long and short ranges. Perhaps the solution to this issue lies in adapting the algorithm itself. The system could also possibly use integrated sensors such as altimeters or range finders to increase accuracy. Regardless, further development and analysis is required to increase the range of the system.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. HSV AND GRAYSCALE PROCESSING TIMES

Trial	Grayscale Time (s)	HSV Time (s)
1	0.168	0.072
2	0.151	0.043
3	0.158	0.040
4	0.195	0.043
5	0.195	0.046
6	0.185	0.050
7	0.190	0.058
8	0.194	0.047
9	0.265	0.053
10	0.192	0.052
11	0.195	0.048
12	0.231	0.055
13	0.192	0.052
14	0.294	0.047
15	0.187	0.048
16	0.178	0.047
17	0.194	0.048
18	0.227	0.054
19	0.184	0.057
20	0.193	0.047
21	0.197	0.047
22	0.189	0.054
23	0.210	0.073

Trial	Grayscale Time (s)	HSV Time (s)
24	0.191	0.046
25	0.192	0.051
26	0.193	0.052
27	0.201	0.046
28	0.204	0.046
29	0.186	0.048
30	0.202	0.048
AVG:	0.198	0.051
STDEV:	0.027536	0.007243
VAR:	0.000758	5.25E-05

LIST OF REFERENCES

AirNav. 2017. “KMRY Monterey Regional Airport.” AirNav, LLC, April 27. Accessed May 24, 2017. http://www.airnav.com/airport/KMRY.

Cambridge in Colour. 2017. “Sharpening using an Unsharp Mask.” Accessed April 19, 2017. http://www.cambridgeincolour.com/tutorials/unsharp-mask.htm.

Cheng, H. D., X. H. Jiang, Y. Sun, and Jingli Wang, 2001. "Color Image Segmentation: Advances and Prospects." *International Journal on Image, Graphics, and Signal Processing* 34, no. 12 (February): 28-34. Accessed May 18, 2017. http://www.sciencedirect.com/science/article/pii/S0031320300001497.

Dangeti, Sarita Veera. 2003. “Denoising Techniques—A Comparison.” Master’s thesis, Louisiana State University. Accessed April 12, 2017. http://etd.lsu.edu/docs/available/etd-1219102-152426/unrestricted/Dangeti_thesis.pdf.

Defense Standardization Program. 2010. “Case Study: Joint Precision Approach and Landing System.” Defense Standardization Program Office: Fort Belvoir, VA. Accessed 15 April, 2017. http://quicksearch.dla.mil/Transient/-E13A9F9381654210B19578B7AE99B036.pdf.

Dougherty, Edward R., and Roberto A. Lotufo. 2003. *Tutorial Texts in Optical Engineering*, Vol. 59, *Hands-on Morphological Image Processing*. Bellingham, WA: SPIE Press. Accessed April 13, 2017. http://ebooks.spiedigitallibrary.org.libproxy.nps.edu/book.aspx?bookid=159.

Duda, Richard O., and Peter E. Hart. 1971. “Use of the Hough Transformation to Detect Lines and Curves in Pictures.” *Communications of the ACM* 15, no. 1 (January): 11–15. Accessed 17 April, 2017. http://www.dtic.mil/docs/citations/ADA457992.

Federal Aviation Administration (FAA). 2013. *Advisory Circular 150/5340-1L: Standards for Airport Markings*. US Department of Transportation. Accessed April 6, 2017. https://www.faa.gov/documentLibrary/media/Advisory_Circular/150_5340_1l.pdf.

———. 2014. “Ground-Based Navigation - Low Power Distance Measuring Equipment (LPDME).” US Department of Transportation, March 25. Accessed May 23, 2017. https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gbng/lpdme/.

———. 2016a. *Airplane Flying Handbook Chapter 8*. US Department of Transportation, Oklahoma City. Accessed May 23, 2017. https://www.faa.gov/regulations_policies/handbooks_manuals/aviation/airplane_handbook/media/10_afh_ch8.pdf.

———. 2016b. “Instrument Landing System (ILS).” US Department of Transportation, December 20. Accessed 23 May, 2017. https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gbng/ils/.

Federal Aviation Administration Southern Region Airports Division. 2015. *A Quick Reference to Airfield Standards*. US Department of Transportation. Accessed April 6, 2017.https://www.faa.gov/airports/southern/airport_safety/part139_cert/media/aso-airfield-standards-quick-reference.pdf.

Fleet Readiness Center Southwest Public Affairs. “North Island Depot Supports New Landing System.” Naval Air Systems Command. Accessed May 23, 2017. http://www.navair.navy.mil/index.cfm?fuseaction=home.NAVAIRNewsStory&id=2489.

Gloria, Jose R. Espinosa. 2016. “Runway Detection from Map, Video, and Aircraft Navigational Data.” Master’s thesis, Naval Postgraduate School. Accessed 19 April, 2017. http://calhoun.nps.edu/bitstream/handle/10945/48516/16Mar_Espinosa_Gloria_Jose.pdf?sequence=1&isAllowed=y.

Golovcsenko, Igor V. 1976. *Computer Simulation of Fresnel Lens Optical Landing System*. Orlando, FL: Naval Training Equipment Center. Accessed May 23, 2017. http://www.dtic.mil/dtic/tr/fulltext/u2/a038456.pdf.

Hough, P. V. C. 1962. Method and Means for Recognizing Complex Patterns. U.S. Patent 3,069,654, filed March 25, 1960, and issued December 18, 1962.

Joblove, George H., and Donald Greenberg. 1978. “Color Spaces for Computer Graphics.” *ACM SIGGRAPH Computer Graphics* 12, no. 3 (August 23): 20–25. Accessed April 11, 2017. http://dl.acm.org/citation.cfm?doid=800250.807502.

Kirsch, R. A., L. Cahn, C. Ray, and G. H. Urban. 1957. “Experiments in Processing Pictorial Information with a Digital Computer.” *IRE-ACM-AIEE ‘57 (Eastern) Papers and Discussions Presented at the December 9–13, 1957,Eastern Joint Computer Conference: Computers with Deadlines to Meet*. New York: Association for Computing Machinery.

Loegering, Greg and Steve Harris. 2002. "Landing Dispersion Results - Global Hawk Auto-Land System." *AIAA’s 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles,* Norfolk, VA. May 20-23, 2003. Accessed May 9,2017. https://arc.aiaa.org/doi/pdf/10.2514/6.2002-3457.

Maini, Raman, and Himanshu Aggarwal. 2009. “Study and Comparison of Various Image Edge Detection Techniques.” *International Journal of Image Processing* 3, no. 1 (March 1): 1–11. Accessed April 19, 2017. https://doaj.org/article/3358147a5a91426fad63f55fa8eacfa8.

MathWorks. 2017a. “imfill.” Accessed April 14, 2017. The MathWorks Inc. https://www.mathworks.com/help/images/ref/imfill.html.

———. 2017b. “bwareaopen.” Accessed April 14, 2017. The MathWorks Inc. https://www.mathworks.com/help/images/ref/bwareaopen.html.

———. 2017c. “medfilt2.” Accessed April 14, 2017. The MathWorks Inc. https://www.mathworks.com/help/images/ref/medfilt2.html.

———. 2017d. “imsharpen.” Accessed April 14, 2017. The MathWorks Inc. https://www.mathworks.com/help/images/ref/imsharpen.html.

———. 2017e. “edge.” Accessed April 14, 2017. The MathWorks Inc. https://www.mathworks.com/help/images/ref/edge.html.

———. 2017f. “hough.” Accessed April 14, 2017. The MathWorks Inc. https://www.mathworks.com/help/images/ref/hough.html.

———. 2017g. “houghpeaks.” Accessed April 14, 2017. The MathWorks Inc. https://www.mathworks.com/help/images/ref/houghpeaks.html

———. 2017h. “houghlines.” Accessed April 14, 2017. The MathWorks Inc. https://www.mathworks.com/help/images/ref/houghlines.html

———. 2017i. “Morphological Dilation and Erosion.” Accessed May 15, 2017. The MathWorks Inc. https://www.mathworks.com/help/images/ref/imfill.html.

Northrop Grumman, 2017. “Global Hawk.” Access 8 May 2017. Northrop Grumman Corporation. http://www.northropgrumman.com/Capabilities/GlobalHawk/Pages/default.aspx.

———. 2015. “X-47B UCAS: Unmanned Combat Air System.” Northrop Grumman Systems Corporation, San Diego. Accessed May 9, 2017. http://www.northropgrumman.com/Capabilities/X47BUCAS/Documents/UCAS-D_Data_Sheet.pdf.

Office of Technical Intelligence. 2015. *Department of Defense Research and Engineering Technical Assessment: Autonomy*. Office of the Assistant Secretary of Defense for Research & Engineering, Washington, DC. Accessed May 23, 2017. http://www.defenseinnovationmarketplace.mil/resources/OTI_TechnicalAssessment-AutonomyPublicRelease_vF.pdf.

Patidar, Pawan, Manoj Gupta, Sumit Srivastava, and Ashok Kumar Nagawat. 2010. "Image De-Noising by various Filters for Different Noise." *International Journal of Computer Applications* 9, no. 4 (November 10): 45–50. Accessed April 8, 2017. http://search.proquest.com/docview/814856118.

Quijas, Jonathan, and Olac Fuentes. 2014. "Removing JPEG Blocking Artifacts using Machine Learning." *2014 IEEE Southwest Symposium on Image Analysis and Interpretation*. San Diego: IEEE.

Shang, Jiajia, and Zhongke Shi. 2007. "Vision-Based Runway Recognition for Autonomous UAV Landing." *IJCSNS International Journal of Computer Science and Network Security* 7 no. 3: 112–117. Accessed April 7, 2017. http://www.sciencedirect.com/science/article/pii/S1877705812021844.

Smith, Alvy Ray. 1978. "Color Gamut Transform Pairs." *ACM SIGGRAPH Computer Graphics* 12, no. 3 (August 23): 12–19. Accessed April 11, 2017. http://dl.acm.org/citation.cfm?id=807361.

Sobel, I., and G. Feldman. 1968. "An Isotropic 3x3 Image Gradient Operator." Lecture, Stanford Artificial Intelligence Project. Accessed April 19, 2017. https://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator.

US Department of Defense. 2011. *Unmanned Systems Integrated Roadmap FY2011-2036*. Office of the Under Secretary of Defense Acquisition, Technology, and Logistics. Washington, DC. Accessed May 23. 2017. http://www.acq.osd.mil/sts/docs/Unmanned%20Systems%20Integrated%20Roadmap%20FY2011-2036.pdf.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California